Topic	JOYSTICK	
Class Description	**Students will learn about the joystick component. Students will also learn how a joystick works. Using this knowledge, students will create a project to animate an LED Dot Matrix display.**	
Class	**PRO C270**	
Class time	**45 mins**	
Goal	• Learning about the joystick component. • Understanding the difference between analog signals and digital signals. • Learning to light up a dot on the LED Dot Matrix display depending on the joystick.	

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** **05 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:** • Greet the student. • Revision of previous class activities. • Quizzes.	**ESR**: Hi, thanks! Yes, I am excited about it! 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:** • Appreciate the student. • Narrate the story by using hand gestures and voice modulation methods to bring more interest in students.	
TEACHER-LED ACTIVITY - 15mins	
Teacher Initiates Screen Share	
• **Understanding the working principle of a joystick.** • **Learning how to read analog data through code.**	
Teacher Action	**Student Action**
Do you remember what we did in the previous class? Do you have any doubts? *If the student has any doubts, clarify the doubts.*	**ESR**: We learned about LED Dot Matrix with MAX7219 controller. **ESR:** Varied.

Have you played video games before?	**ESR:** Yes. I love video games.
In the video game controllers, have you ever noticed something called a joystick?  Student can view this video in Student Activity 1.	**ESR:** Yes. Joysticks are usually used to control the character's / vehicle's direction of movement.

Yes. Exactly. Joysticks are also used in aircraft to control various aspects of the flight.

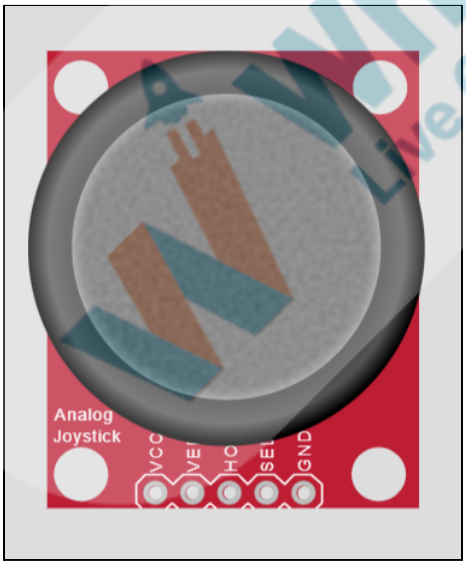
Have you seen this component being used anywhere else?

ESR: varied.

Great!

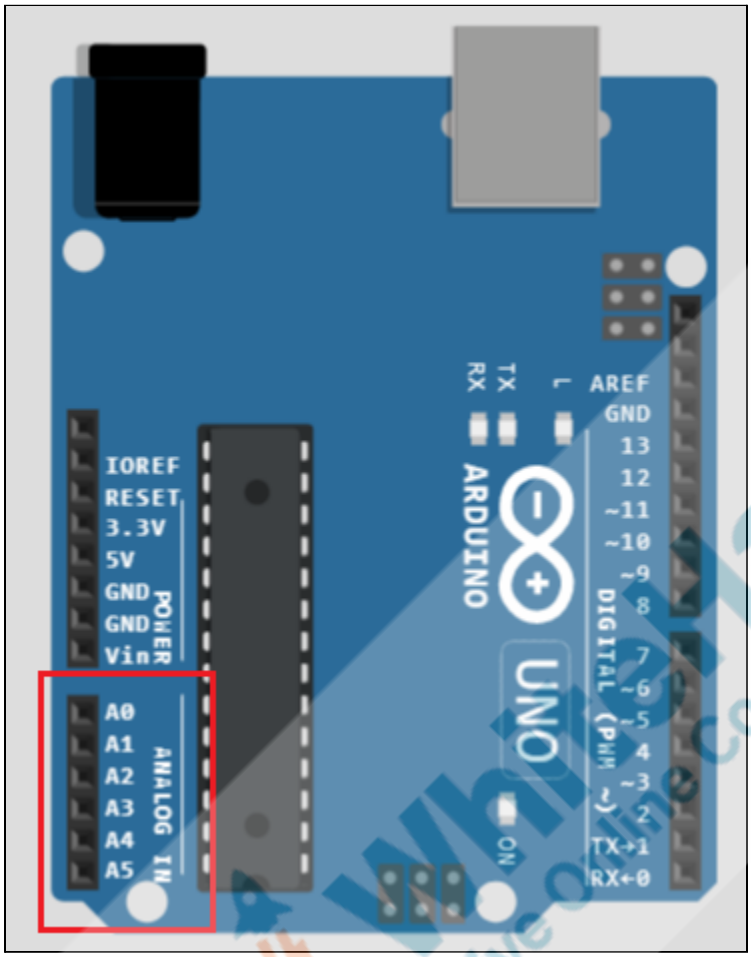
We can connect a joystick to our Arduino Uno board as well. Let's try to understand how it works.

Teacher adds a new joystick component in wokwi and shows it to the student. Student observe the component.



Did you notice that the component name is analog joystick? Let's try to understand what that means. Till now, we have worked with digital signals where the states are HIGH or LOW. But sometimes the values of a signal are in a range instead of just being ON/OFF. Think about light, radio waves, sound waves. It's not either in ON or OFF state. It is continuous. These are all analog signals. As we understand analog and digital signals now, let's observe the output of the joystick.	**ESR:** Yes.
Let's look at the pins of the joystick component. **Pin Name** / **Description** VCC — Voltage supply VERT — Vertical axis output (analog) HORZ — Horizontal axis output (analog) SEL — Push button GND — Ground	

We know VCC and GND already. Let's understand the other three- VERT- represents how much we have moved the joystick on the y-axis / vertical axis. It is an analog pin. HORZ- represents how much we have moved the joystick on the x-axis / horizontal axis. It is an analog pin. SEL- represents the pin to control the push button. This is a digital pin.	
Let's design the circuit first. For our project, we are going to open wokwi.com and start a new Arduino Uno project. 1. Components: ● 1 x **Arduino Uno board** ● 1 x **Analog Joystick** 2. Let's do connections: First, let's observe the Arduino Uno board. We can see that there is a separate section for Analog pins	



All the analog inputs need to be connected to this section.

In a joystick, the **HORZ** and **VERT** pins give us analog value. So, we connect those 2 pins here.

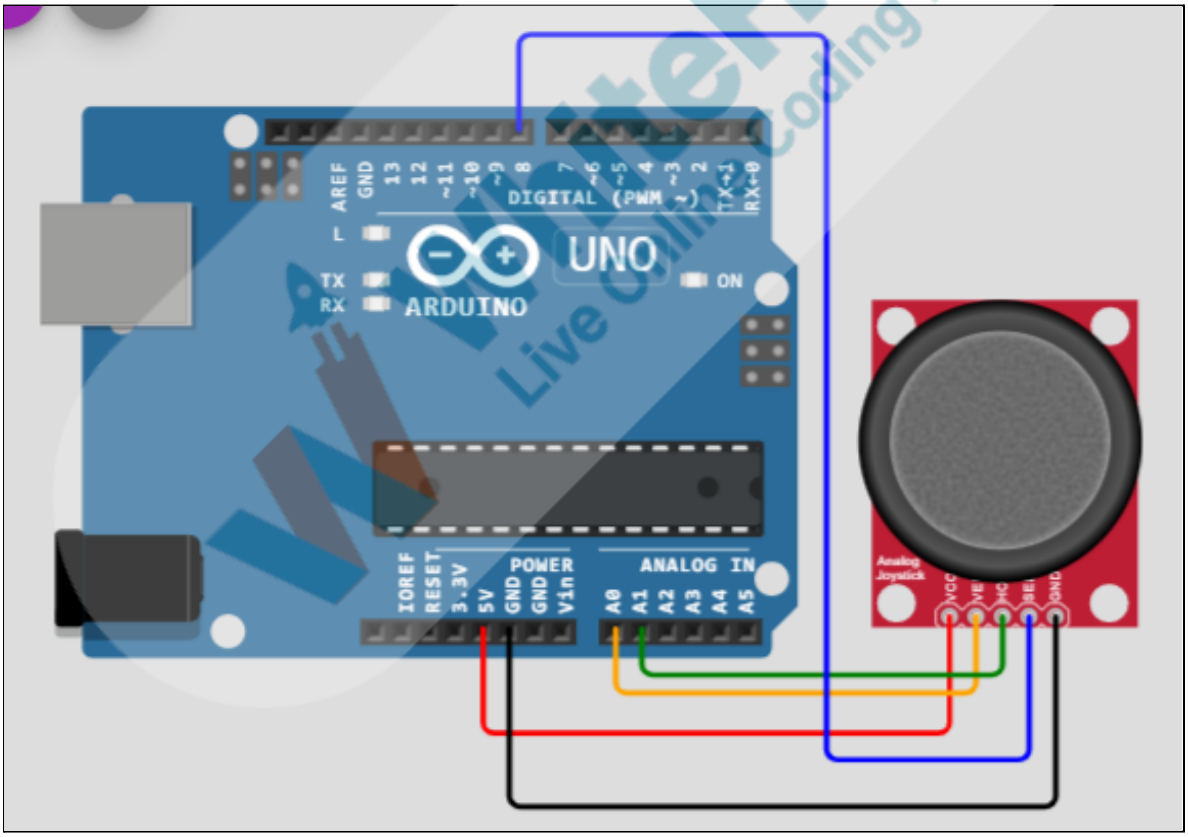
The **SEL** pin gives a digital output so we connect it to GPIO 8.

Arduino Uno board	Analog Joystick Pin Number
5V	VCC
Ground	GND

A0	VERT
A1	HORZ
GPIO 8	SEL

*Note: Wire color can be changed by **clicking over it** and selecting the color, or via **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:



Now it's time to write the code.

1. First, go to **sketch.ino.** Define two pins **vpin** and **hpin** and assign A0 and A1 to it-

```
const byte vpin = A0;

const byte hpin = A1;
```

2. Now, let's write the **setup()** method.

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(25);
}
```

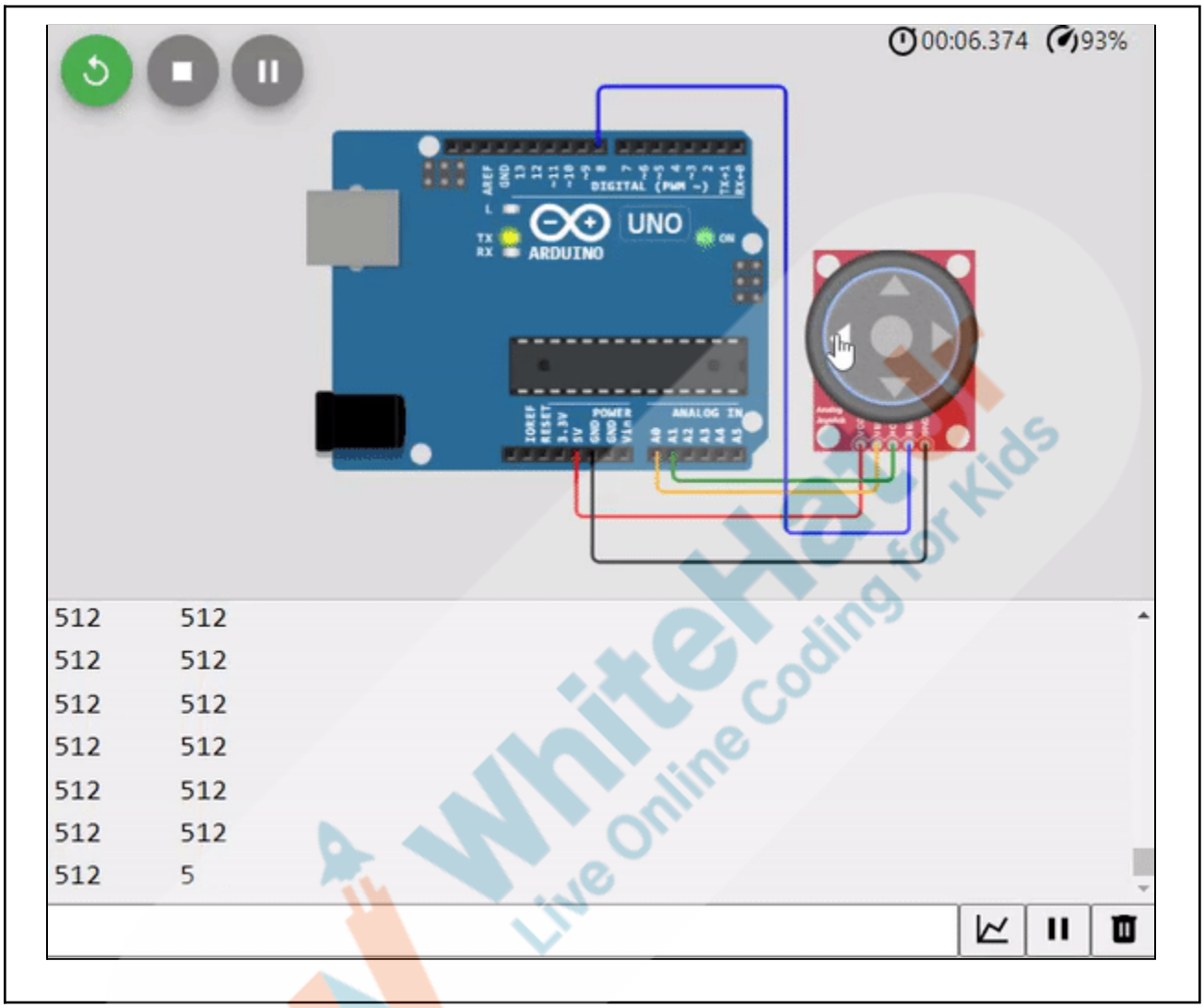
3. We use the function **analogRead()** to read values from the analog pins. We print it in the Serial monitor to observe the output.

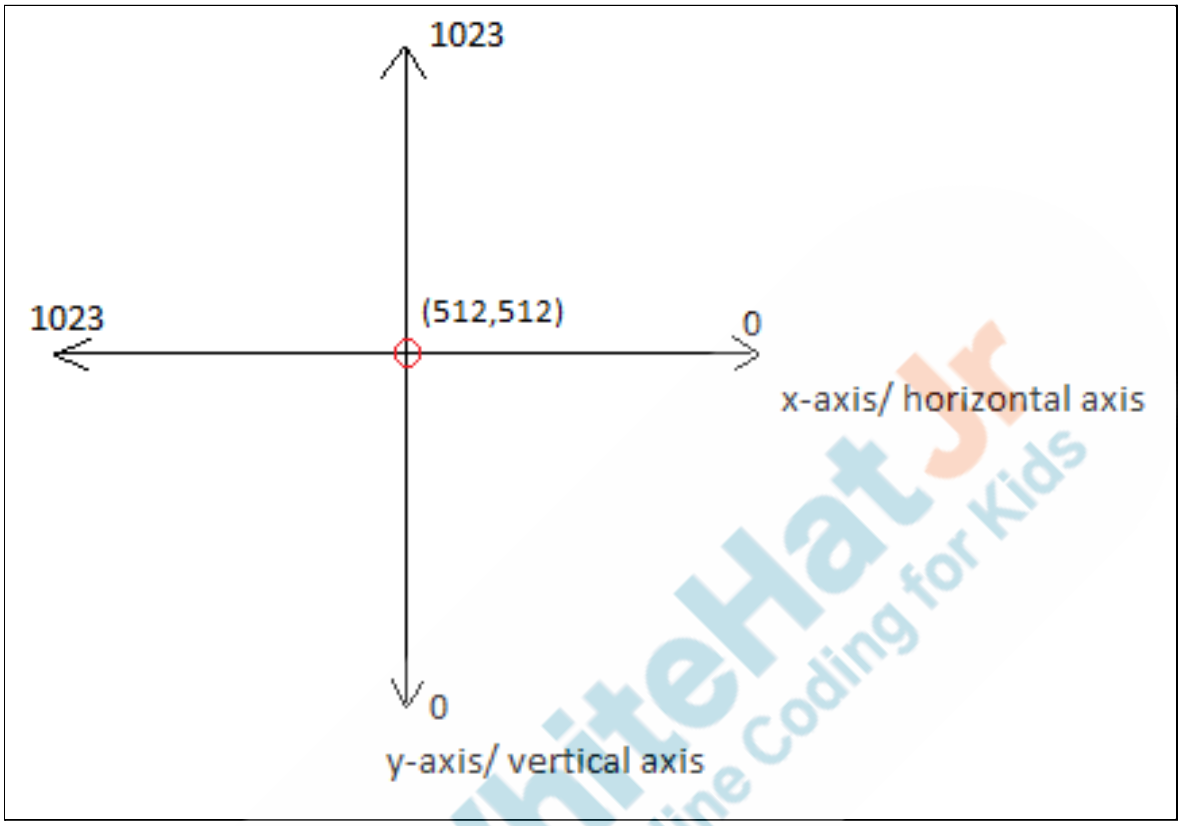
```
void loop() {
  // put your main code here, to run repeatedly:
  Serial.print(analogRead(hpin));
  Serial.print('\t');
  Serial.println(analogRead(vpin));
}
```

Your code should look like this -

```
const byte vpin = A0;
const byte hpin = A1;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(25);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.print(analogRead(hpin));
  Serial.print('\t');
  Serial.println(analogRead(vpin));
}
```

Let's observe the output-





Now, we can see that the value ranges between 0 to 1023 i.e. total 1024 values or 2^{10} values. Here,

When the joystick is at center the values are (512,512).

If it's on the left, the values are - (1023,512).

If it's on the right, the values are - (0,512).

If it's on the up, the values are - (512,1023).

If it's on the down, the values are - (512,0).

We can see that instead of being HIGH/LOW, the values are in a range of 0-1023. That's why it is an analog signal.

So today with the joystick, we will control the animation pattern on a LED Dot Matrix

<table>
<tr><td colspan="2">display. Let’s get started with the code.</td></tr>
<tr><td colspan="2">As you understood the basics. You will build a project to display.</td></tr>
<tr><td colspan="2">Teacher Stops Screen Share</td></tr>
<tr><td>Can you solve it?

Let's try. I will guide you through it.</td><td>ESR: Yes, sure!</td></tr>
<tr><td colspan="2">STUDENT-LED ACTIVITY- 15 mins</td></tr>
<tr><td colspan="2">• 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.</td></tr>
<tr><td colspan="2">Student Initiates Screen Share</td></tr>
<tr><td colspan="2">ACTIVITY

• Build a program to light up a dot/LED on the LED Dot Matrix display depending on the joystick.</td></tr>
<tr><td>Teacher Action</td><td>Student Action</td></tr>
<tr><td>Teacher guides the student to download the boilerplate code from Student Activity 3</td><td>Student opens wokwi simulator.</td></tr>
<tr><td>Let’s try to create the circuit diagram first. The LED Dot Matrix unit is already added in the boilerplate code.

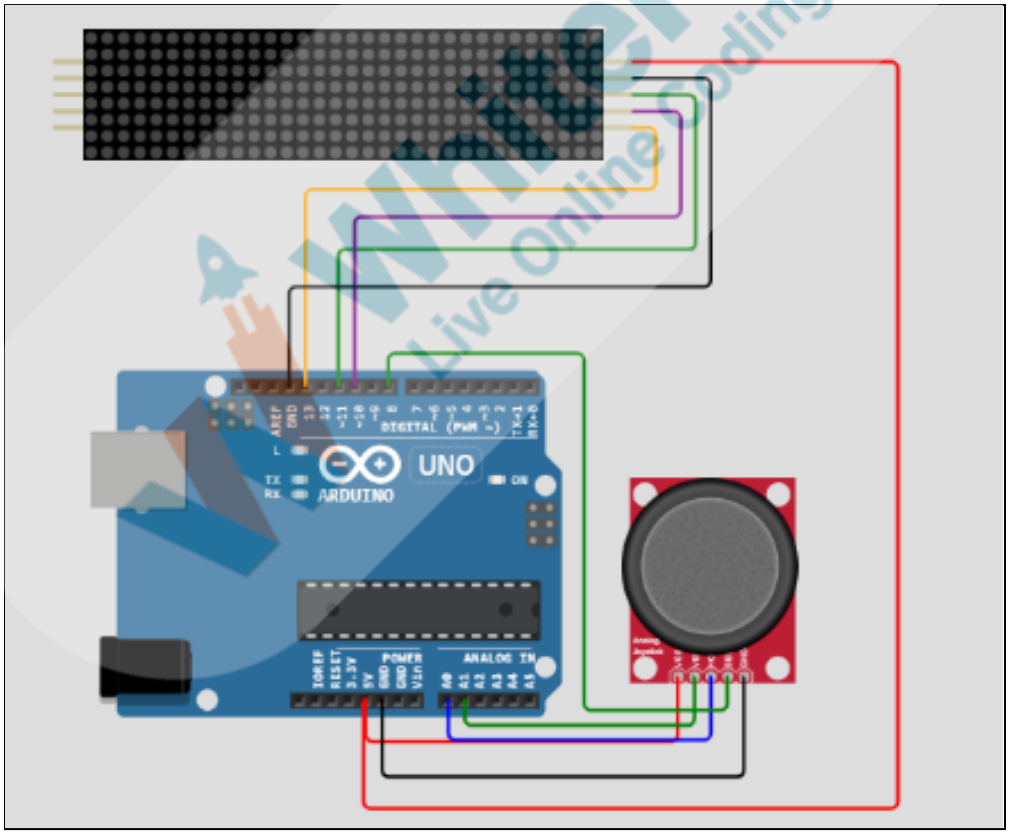
1. Add the following components-

• 1x Arduino Uno board
• 1x Analog Joystick

2. Connections-</td><td></td></tr>
</table>

Arduino Uno board	Analog Joystick Pin Number
5V	VCC
Ground	GND
A0	VERT
A1	HORZ
GPIO 8	SEL

Reference circuit:



Let's work on the code now.

1. First, go to **sketch.ino.** Let's define 5 variables - **head_x**, **head_y**, **direction**, **prev_ direction**, **max_x** and **max_y**. Here, **head_x** and **head_y** will hold the joystick's position. We will assign the direction variable a value depending on the joystick's position. The **max_x** and **max_y** variable will hold the maximum column and row number respectively for the LED Dot Matrix display.

```
int head_y = 0;
int head_x = 0;

String direction="" , prev_direction = "";
int max_x = 31, max_y = 7;
```

2. Assign the analog pin numbers to the **hpin** and **vpin** constants. These should be the analog pin numbers to which you have connected the **HORZ** and **VERT** pins of the **Analog Joystick**

```
// define the joystick controls
const byte hpin = A0;
const byte vpin = A1;
```

3. Now, let's write the following code inside the **setup()** function.
   - Initiate the module.

```
matrix.begin();
```

   - Clear the Dot Matrix display, if there is anything –

```
matrix.clear();
```

- Use the setPoint() function to light up the x and y position on the LED.

  ```
  matrix.setPoint(head_y, head_x, true);
  ```

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

  matrix.begin();
  matrix.clear();

  matrix.setPoint(head_x,head_y,true);

}
```

4. Let's define a new function named **check_direction().**

   This function will read the horizontal and vertical positions of the joystick. Depending on these values, we will assign a value to the direction variable.

   - First, write an **if** statement. In the first condition, we will read the hpin value of the joystick. If it is more than 512, we will assign the direction as "**left**".
   - Now, if the hpin value of the joystick is less than 512, we will assign the direction as "**right**".
   - Similarly, if the vpin value of the joystick is more than 512, we will assign the direction as "**up**".
   - And if the vpin value of the joystick is less

than 512, we will assign the direction as "**down**".

```
void check_direction(){

  if(analogRead(hpin)>512)
    direction="left";
  else if(analogRead(hpin)<512)
    direction="right";
  else if(analogRead(vpin)>512)
    direction="up";
  else if(analogRead(vpin)<512)
    direction="down";

}
```

5. Now, we will write another function named **move_sprite().**

   This function will check the **direction** variable and will change **x** and **y** variables accordingly.

   - First, write an **if** statement. In the first condition, we will check the direction variable. If its value is "**left**", we will increase x by 1.
   - Now, we will check the direction variable again. If its value is "**right**", we will decrease x by 1.
   - Similarly, we will check the direction variable. If its value is **"down**", we will increase y by 1.
   - Now, we will check the direction variable again. If its value is "**up"**, we will decrease y by 1.

```
void move_sprite(){

  if (direction  ==  "left")  head_x++;
  else if (direction  ==  "right")  head_x--;
  else if (direction  ==  "up") head_y--;
  else if (direction  ==  "down") head_y++;

}
```

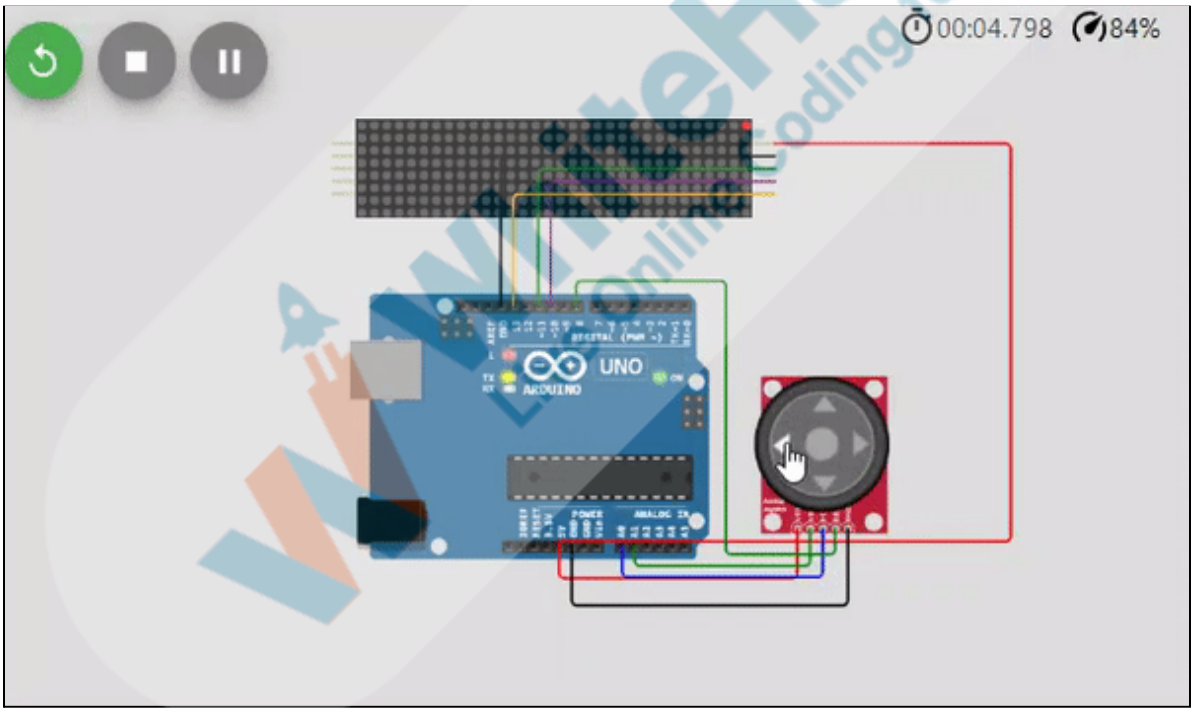
6. Let's call the **check_direction()** and **move_sprite()** in the **loop()** method now.

```
void loop(){


  check_direction();
  move_sprite();

}
```

7. Now, map the **head_x**, **head_y** position on the LED Dot Matrix display.

```
void loop(){


  check_direction();
  move_sprite();

  matrix.setPoint(head_y, head_x, true);
}
```

Let's observe the output now.



Click here to view the reference video.

Did you observe any problem here?	**ESR:** It lights up the next LEDs very fast. We want it

<table>
<tr>
<td></td>
<td>to be a little slower.</td>
</tr>
<tr>
<td>How can we do that?

Perfect! Let's do that.</td>
<td>ESR: We can add the delay() method.</td>
</tr>
<tr>
<td>8. Let's add a delay() of 200 milliseconds.

<pre>
void loop(){

  check_direction();
  move_sprite();

  matrix.setPoint(head_y, head_x, true);
  delay(200);
}
</pre>
9. Now, do we notice any other problems?

Teacher lets the student observe the code and figure out the problem.

Exactly!

Let's resolve this problem.

In our case, an LED Dot matrix has 8x8 LED.

Vertically- the y value will range between 0 to 7.</td>
<td>ESR: The x, y increases or decreases infinitely. So, it goes out of the LED Dot Matrix.</td>
</tr>
</table>

Horizontally- the x value will range between 0 to 31 as we have daisy-chained 4 LED Dot Matrix units.

But, we want to make sure that when the head_x reaches any boundary, it should come out of the opposite side. So, if it crosses the last column from the left, it should enter from the right.

Now, write a new function named **window_check()**

```
void window_check(){

  //  window exceed check
  if (head_x > max_x)head_x = 0;
  else if (head_x < 0)head_x = max_x;
  else if (head_y > max_y)head_y = 0;
  else if (head_y < 0)head_y = max_y;

}
```

10. Let's call the **window_check()** function at the end of the **move_sprite()** function.
11. Run the game, give it a try.

12. Let's say if the movement of the sprite is towards "**left**", the next movement should either be "**up**" or "**down**". It should not go back to the exact opposite direction, then we won't be able to observe the movement.

    To resolve this problem, at the end of the **check_direction()** method, we will store the **direction** in the variable named **prev_direction**.

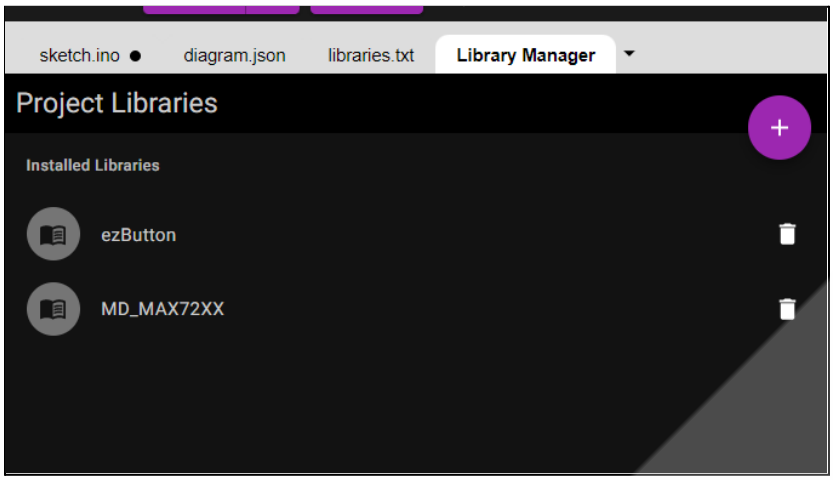
```
void check_direction(){

  if(analogRead(hpin)>512)
    direction="left";
  else if(analogRead(hpin)<512)
    direction="right";
  else if(analogRead(vpin)>512)
    direction="up";
  else if(analogRead(vpin)<512)
    direction="down";

  //  updating prev_direction
  prev_direction = direction;

}
```

13. Let's add the conditions now,

```
void check_direction(){

  if(analogRead(hpin)>512 &&  prev_direction  !=  "right")
    direction="left";
  else if(analogRead(hpin)<512 &&  prev_direction  !=  "left")
    direction="right";
  else if(analogRead(vpin)>512 &&  prev_direction  !=  "down")
    direction="up";
  else if(analogRead(vpin)<512 &&  prev_direction  !=  "up")
    direction="down";

  //  updating prev_direction
  prev_direction = direction;

}
```

Now, we also need to write the code to stop the animation once the push button on the joystick is pressed.

1. To do that, first let's import a header file named "**ezButton.h**"
    - First, go to the **Library Manager** and import the **ezButton** library.



- Now, in the **sketch.ino** file, include the **ezButton.h** file.

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

- Let's initiate an ezButton object that attaches to pin 8.

```
ezButton button(8);
```

2. Initiate a new variable named **flag** to 0. We will change the flag value to 1, once the PUSH button is pressed.

diagram.json | libraries.txt | **sketch.ino** ● | wokwi-project.txt | Library Manager

```
#include <MD_MAX72xx.h>
#include <ezButton.h>

//  matrix controls
const byte data_pin = 11;
const byte chip_select_pin = 10;
const byte clock_pin = 13;
const byte max_devices = 4;

ezButton button(8);

int head_y = 0;  //  initial position (dont use byte, cant use constrain method)
int head_x = 0;
int flag=0;

int max_x = 31, max_y = 7;

String direction="" , prev_direction = "";

//  creating matrix object
MD_MAX72XX matrix = MD_MAX72XX(MD_MAX72XX::PAROLA_HW, chip_select_pin, max_devices);

//  joystick controls
const byte hpin = A0;
const byte vpin = A1;
```

3. Call **button.loop()** method inside the **loop()** method. We must call this method to make the button work.

```
void loop(){

  button.loop();
```

4. Set the debounce time to 25 milliseconds.Pushbuttons generate spurious transitions when pressed. These transitions may be read as multiple presses in a very short time fooling

the program. To debounce a pushbutton is to check twice in a short period of time to make sure the pushbutton is definitely pressed.

```
void setup(){

  Serial.begin(9600);

  matrix.begin();
  matrix.clear();

  button.setDebounceTime(25);
  matrix.setPoint(head_y, head_x, true);

}
```

5. Now, Let's write the code to change the flag input to 1 , once the button is pressed.

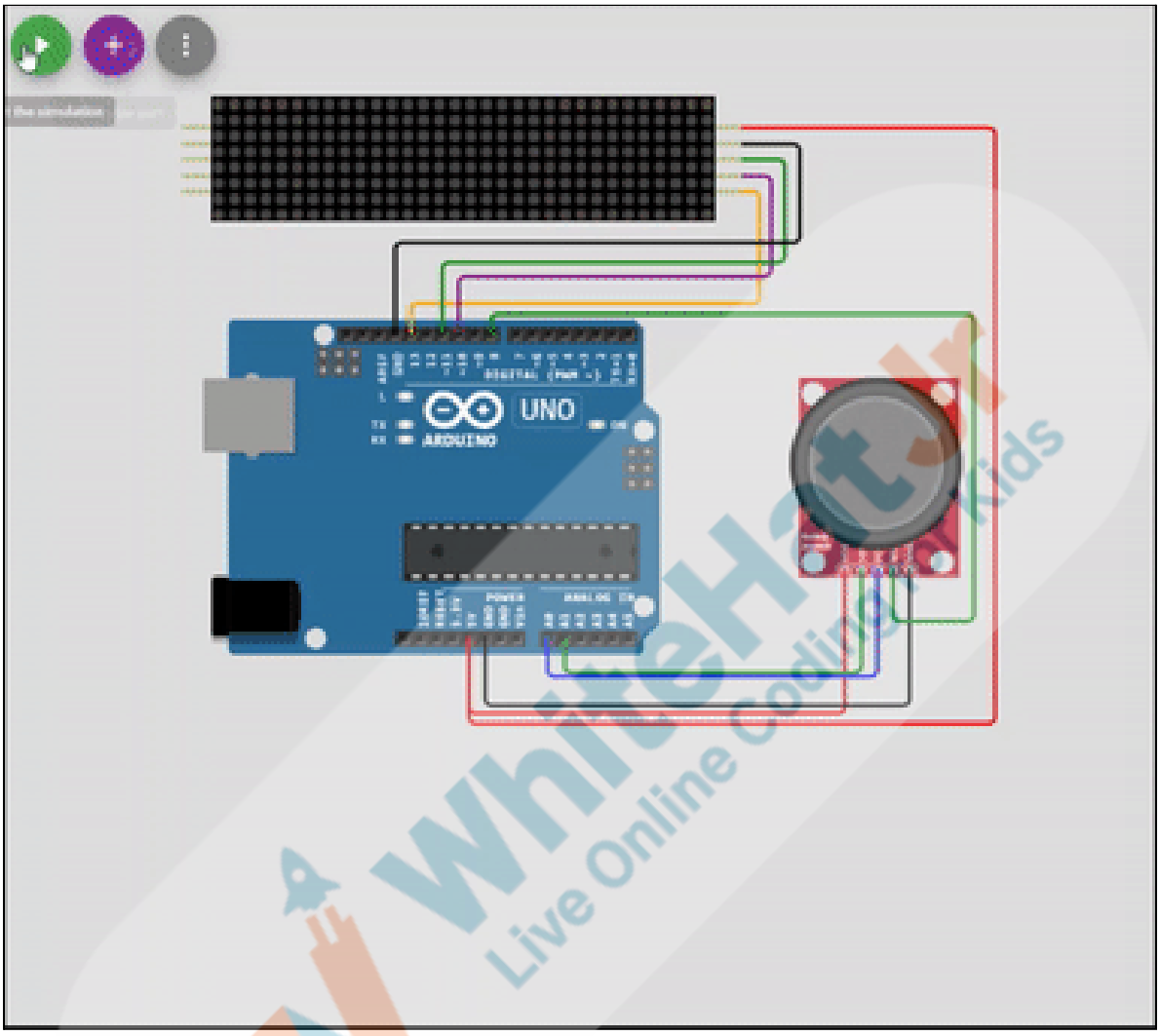
```
void move_sprite(){

  if (direction  ==  "left")  head_x++;
  else if (direction  ==  "right")  head_x--;
  else if (direction  ==  "up") head_y--;
  else if (direction  ==  "down") head_y++;

  window_check();

  if(button.isPressed()){
    Serial.println("The button is pressed");
    flag=1;
  }
}
```

6. Now, we want to run the **check_direction()** and

move_sprite() only when the flag is 0. So, let's add an **if** statement.

```
void loop(){
  button.loop();

  if (flag==0)
  {
    check_direction();
    move_sprite();
  }

  //matrix.clear();
  matrix.setPoint(head_y, head_x, true);
  delay(200);
}
```

Reference Output:



Click here to view the reference video.

We have completed our project for today! Great job!	

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 05 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

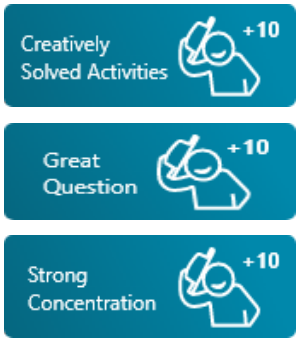
Activity Details

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 build a snake game with the joystick.	*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	https://wokwi.com/
Teacher Activity 2	Read more about joystick	https://www.arduino.cc/reference/en/libraries/joystick/
Teacher Reference	Reference Code	https://github.com/procodingclass/PRO-C270-Reference-Code
Teacher Reference 2	Project	https://s3-whjr-curriculum-uploads.whjr.online/13ac3c32-801c-43a9-ab8a-671c67654802.pdf
Teacher Reference 3	Project Solution	https://github.com/procodingclass/PRO-C270-Project-Solution
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/3884c8fc-8964-4fab-99af-9febf2f60b64.pdf
Student Activity 1	Joystick example	https://s3-whjr-curriculum-uploads.whjr.online/d59e0d7a-cf5f-4777-bcaf-7714a39b60e3.gif
Student Activity 2	Simulator	https://wokwi.com/
Student Activity 3	Boilerplate code	https://github.com/procodingclass/P

		RO-C270-Student-Boilerplate
Student Activity 4	Read more about joystick	https://www.arduino.cc/reference/en/libraries/joystick/