| Topic | PERSONAL TELEVISION - 1 | |
|---|---|---|
| **Class Description** | **Students will learn about PAL TV. Students will also learn how to connect a PAL TV to an Arduino UNO. They will learn to display text and images on the TV.** | |
| **Class** | **PRO C276** | |
| **Class time** | **50 mins** | |
| **Goal** | ● Understand about PAL TV.<br>● Learn to connect a TV to an Arduino board.<br>● Understand the basics of bitmap.<br>● Learn to display images using bitmap.<br>● Learn to display text on the TV. | |
| **Resources Required** | ● Teacher Resources:<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen<br>○ Smartphone<br><br>● Student Resources:<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen | |
| **Class structure** | **Warm-Up**<br>**Teacher-Led Activity**<br>**Student-Led Activity**<br>**Wrap-Up** | **10 mins**<br>**15 mins**<br>**15 mins**<br>**10 mins** |

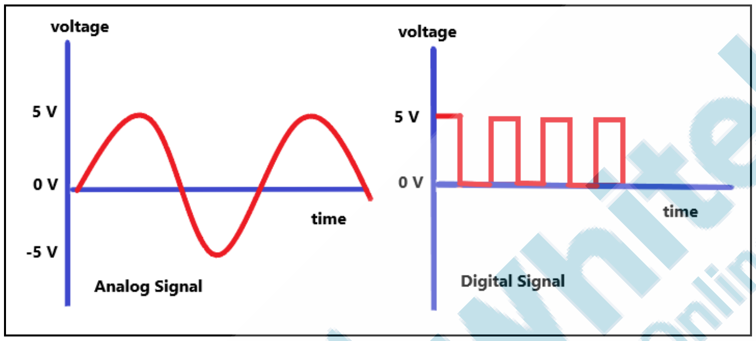
**WARM-UP SESSION - 10 mins**

| Teacher Action | Student Action |
|---|---|
| Hello &lt;student's name&gt;. How are you? It's great to see you! Hope you had a great day so far. | **ESR**: Varied. |

WhiteHat Jr
Live online Coding for Kids

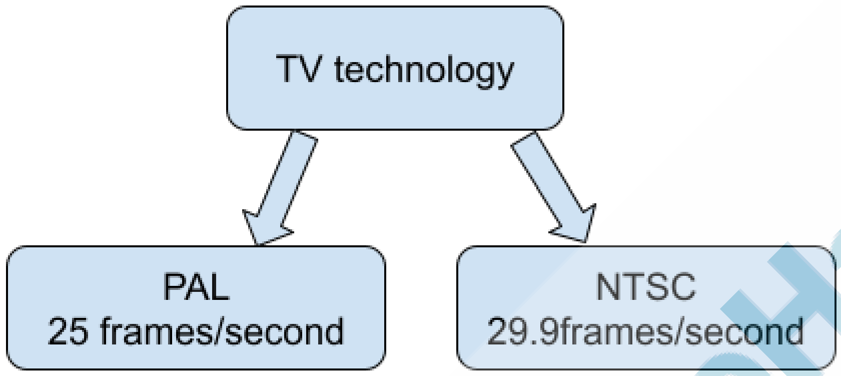
| | |
|---|---|
| **Following are the WARM-UP session deliverables:**<br>● Greet the student.<br>● Revision of previous class activities.<br>● Quizzes. | Click on the slide show tab and present the slides |
| **WARM-UP QUIZ**<br>Click on In-Class Quiz | |
| **Activity Details**<br><br>**Following are the session deliverables:**<br>● Appreciate the student.<br>● Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students. | |
| **TEACHER-LED ACTIVITY 15mins** | |
| **Teacher Initiates Screen Share** | |
| ● **Understand about PAL TV.**<br>● **Learn to connect a TV to an Arduino board.**<br>● **Use bitmaps to display images.** | |
| Teacher Action | **Student Action** |

| | |
|---|---|
| Do you remember what we did in the previous class?<br><br>Can you tell me about the working of a remote control ?<br><br>That's wonderful, you remember the concepts of the previous class very well.<br><br>*If the student has any doubts, clarify the doubts.* | **ESR**: We learned about IR receivers and remote controls.<br><br>**ESR:** A remote control sends InfraRed signals to a device to communicate with it. When we press different buttons on the remote control, it sends different signals. Each button has a unique code so that we can identify each button separately. We lit up the **RGB LED** with different colors for different buttons on the remote control. |
| Can you name a few devices at home that can be controlled by a remote?<br><br>Yes, all these devices can be controlled by a remote control.<br><br>In the last class we learned how to connect a remote to the Arduino UNO board. Today we will learn how to connect a | **ESR:** Television, air conditioner, music system, sometimes lights too. |

| | |
|---|---|
| TV to the Arduino UNO board. We will also learn to display images and text on TV using the Arduino UNO.<br><br>But first let's understand how a TV works.<br><br>There are two types of TV signals, analog and digital.<br><br>We had talked about these signals in one of the previous classes. Do you remember the major difference between analog and digital signals ?<br><br> <br><br>You are right, analog signals are continuous electrical signals. Analog television technology uses analog signals to transmit video and audio. In an analog television broadcast, the brightness, colors and sound are represented by amplitude, phase and frequency of an analog signal. | **ESR:** Analog signals have continuous electrical signals, while digital signals have non-continuous electrical signals. |
| Today we will connect a TV component to an Arduino board. We will use the Arduino board to send electrical signals in a specific format so that images are displayed on the TV component.<br><br>The TV component that we are connecting to the circuit is called **PAL TV**. PAL, which is short for **Phase Alternating** | |

**Line (PAL)** is a color encoding system for analog television.

PAL and NTSC are two different types of video formats. PAL was used in European countries and NTSC was used in the USA, Japan and a few other countries.



A PAL picture is made up of 625 interlaced lines and is displayed at a rate of 25 frames per second. An NTSC picture is made up of 525 interlaced lines and is displayed at a rate of 29.97 frames per second.

Today we will connect a PAL TV to the Arduino board and display images and text.

Isn't it exciting ?

**ESR:** Yes!

Let's design the circuit first.

For our project, we are going to open a project on [wokwi.com](wokwi.com).

*Note: Teacher opens [Teacher Activity 1](Teacher Activity 1) and starts explaining the circuit*

1. <u>Components:</u>
    - 1 x **Arduino Uno board**

- 1 x **PAL TV**

2. **Let's do the connections:**

   The circuit of this project consists of an **Arduino Uno** board and a **PAL TV**

| **PAL TV** | **Arduino Uno PIN** |
|---|---|
| GND | GND |
| VIDEO | 7 |
| SYNC | 9 |

The SYNC pin generates a clock signal so that the video data that is sent to the television is synchronized (sent at equal intervals of time).

Arduino UNO gives just 5V whereas the PAL TV needs 220V of power. In the simulation, the power supply to the PAL TV is supplied automatically. So we need not connect any pins of the UNO board to power up the TV.

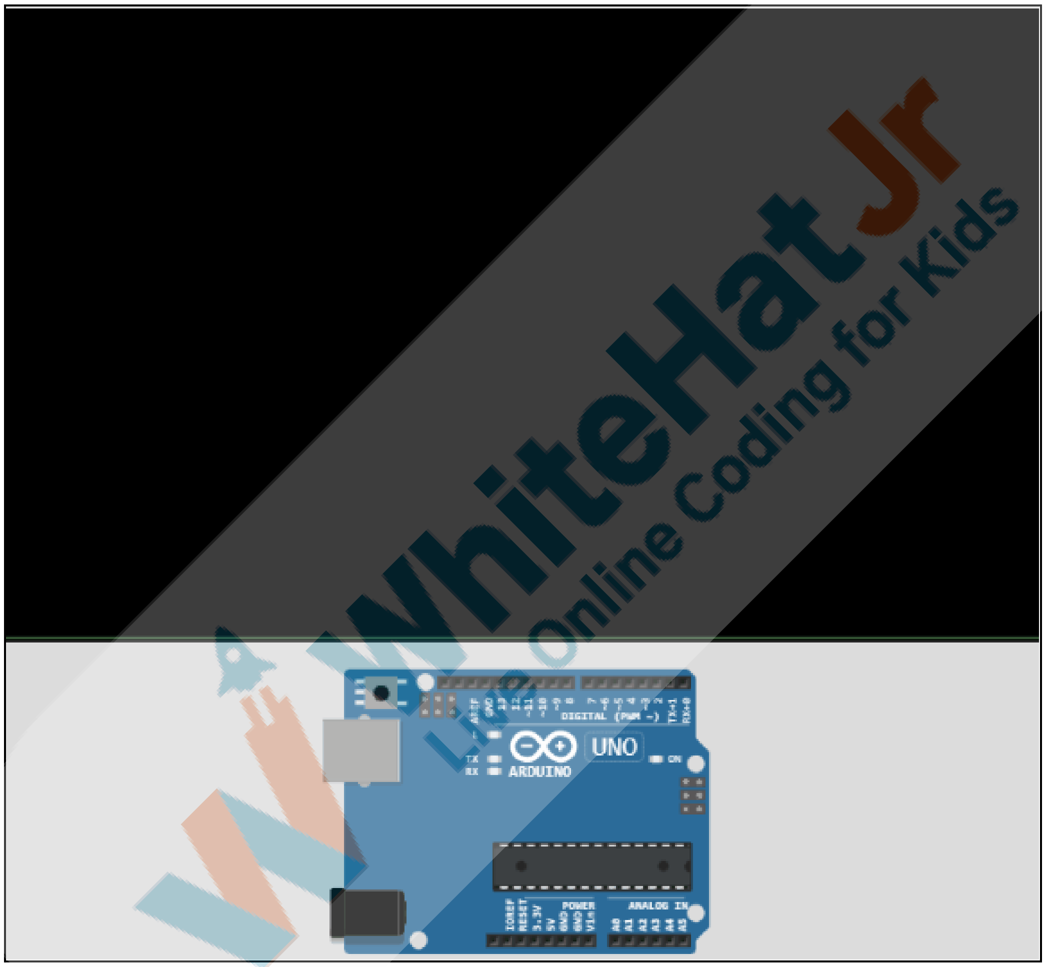
When we make a similar circuit using hardware, (not on a simulator like wokwi), we connect the Arduino with the coaxial cables of the television (the yellow wire or the composite video wire).

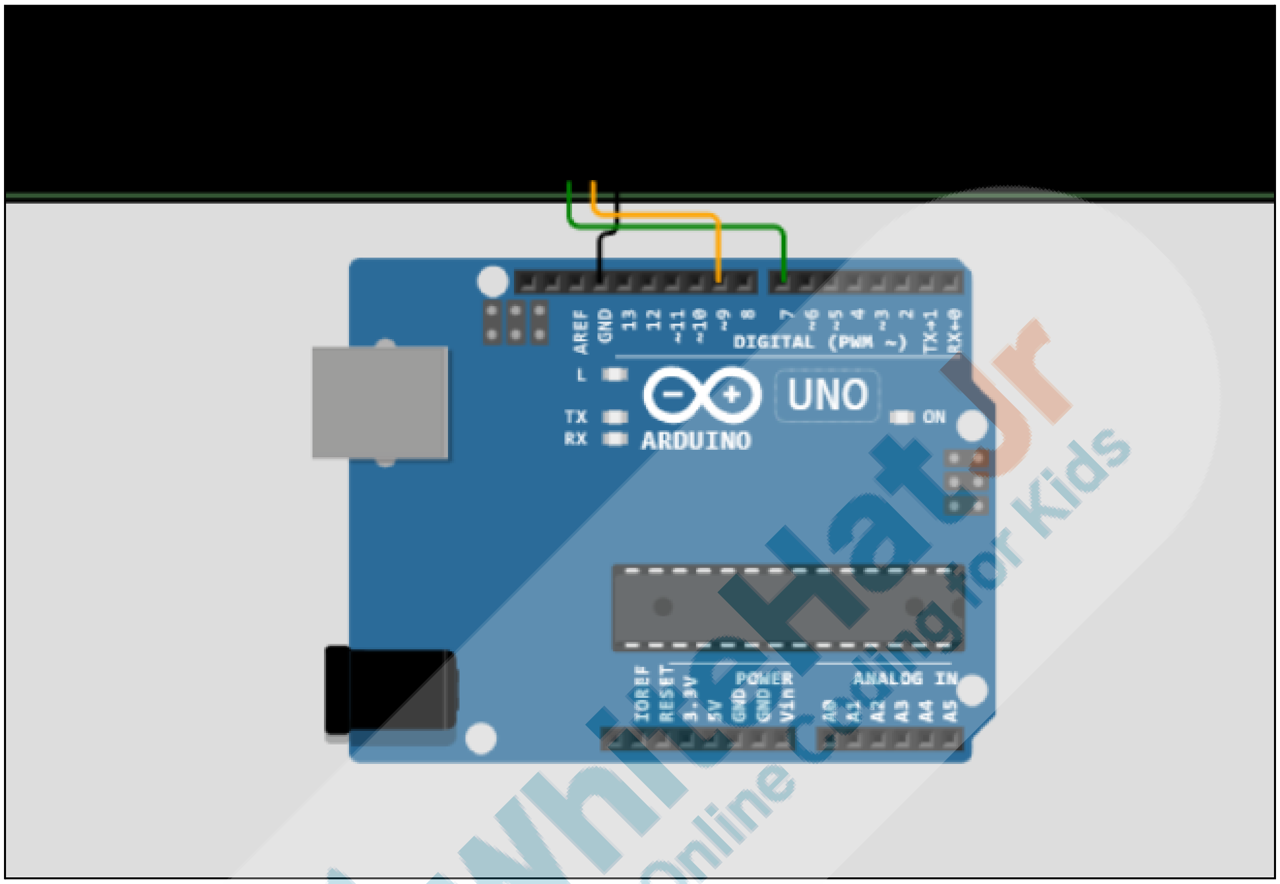
The PALTV that we connect today is a black and white TV and not a color TV.
Additionally, if you want to connect a buzzer to your circuit, you can use pin 11 to connect audio.

*Note: The pins on the PAL TV can be found at the bottom center position of the TV by hovering the mouse in that location.*
*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 track of the component and change the design settings.* | |

Reference:

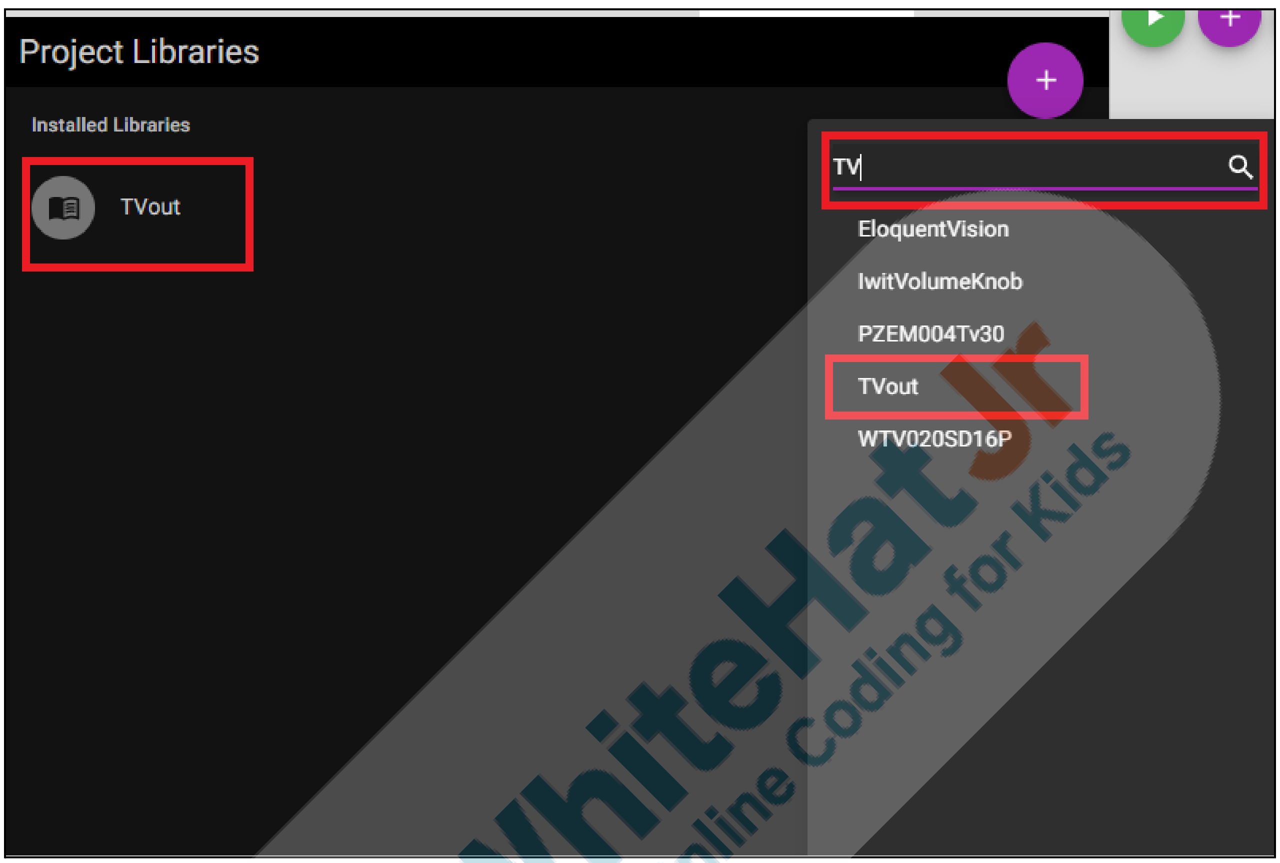




| | |
|---|---|
| Now let's start coding:<br><br>Since we have added a new component in the circuit, can you tell me what needs to be done to support the component before we write the first line of code? | **ESR:** Add library that supports the component. |

Exactly.

1. First, we need to go to the **Library Manager** and **Add a new library** called **TVout**. Once we add the Library, a new file named **libraries.txt** will automatically be created.



2. Now, open **sketch.ino** and the header file at the beginning.

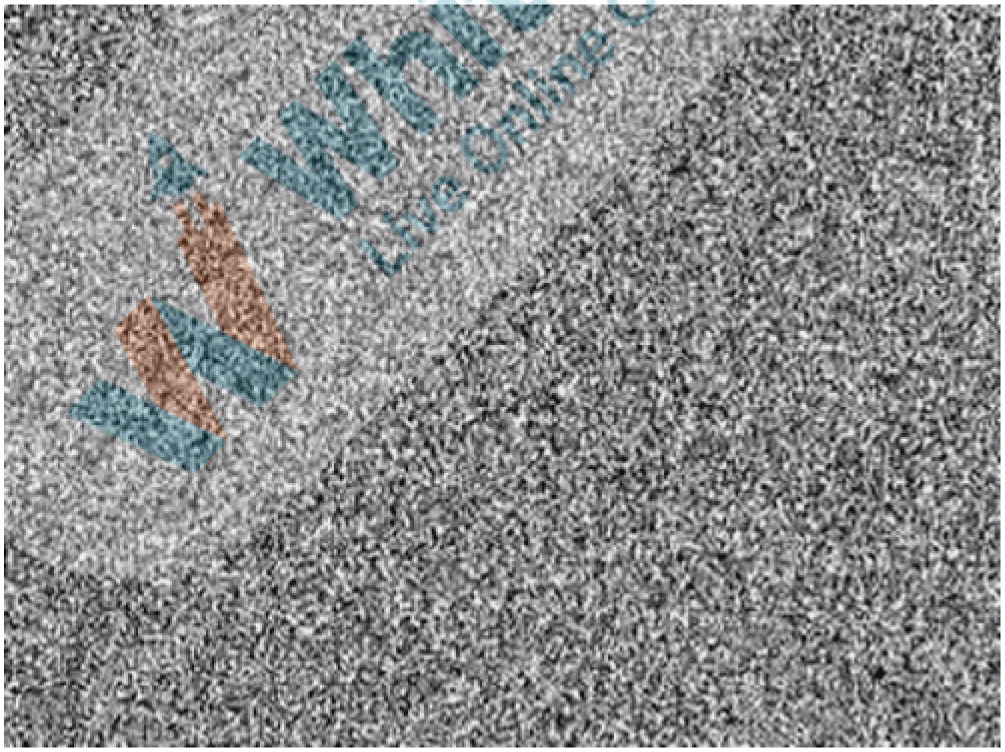
```
#include <TVout.h>
```

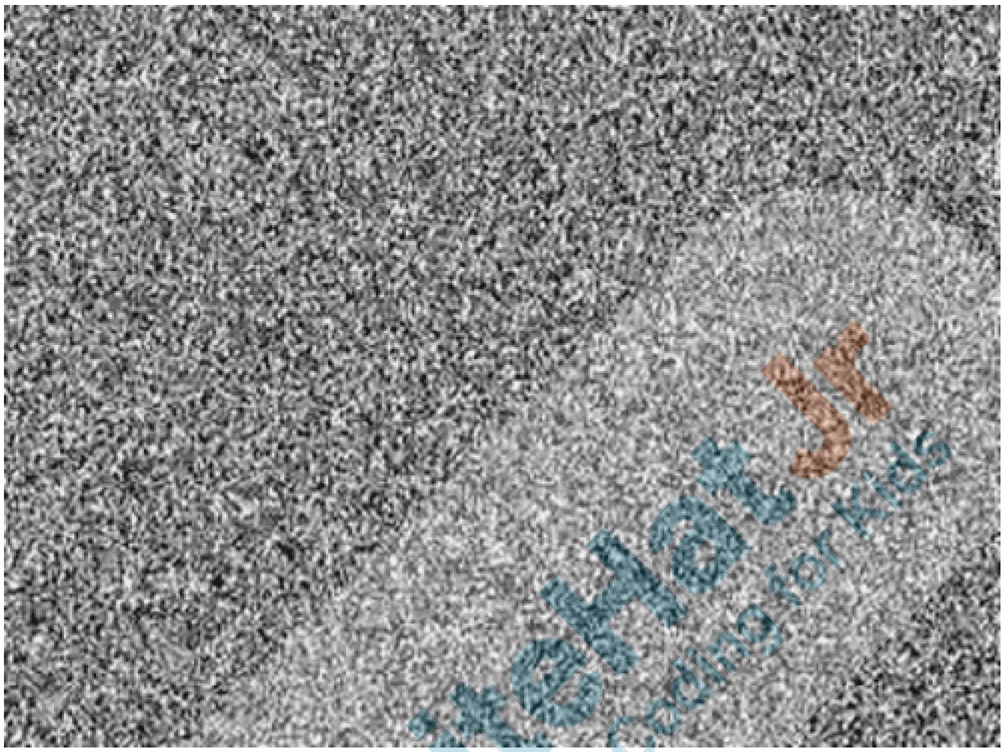
3. Make an object of TVout.

```
TVout tv;
```

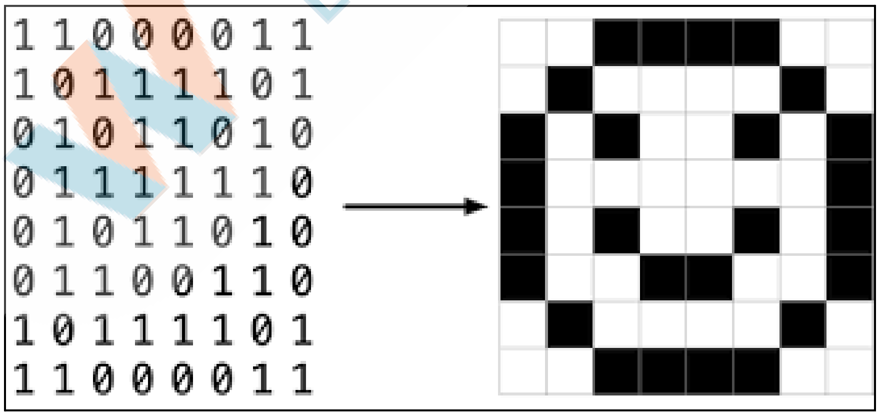
4. In an analog TV,  you would have seen some static video before the program started or before a channel was switched on. Today we will first display the static video on the TV.

5. The static video is displayed as two images one after the other.



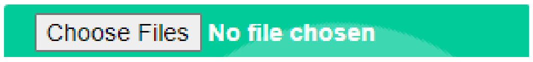


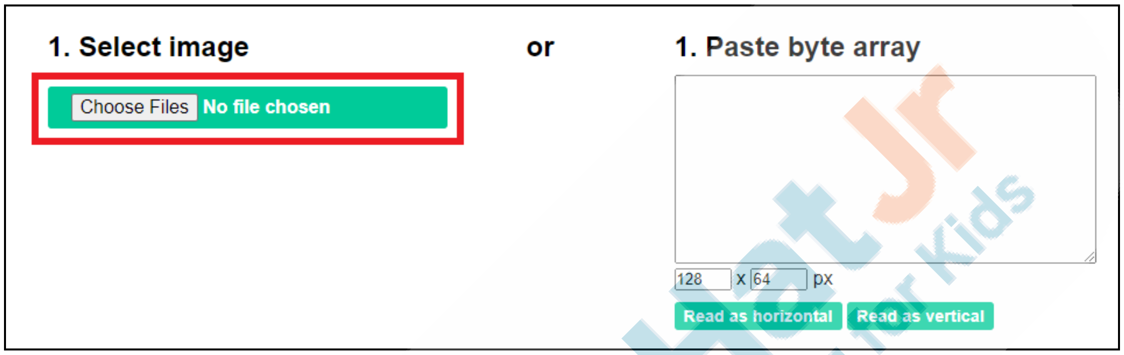
6. To display an image on the TV, it is first converted to a bitmap. A bitmap can be considered a map of bits where each pixel of the image would be either black or white. A bitmap is the representation of the image in **0**s and **1**s. 1 represents white and 0 represents black. By using a combination of black and white we can draw images on a screen.



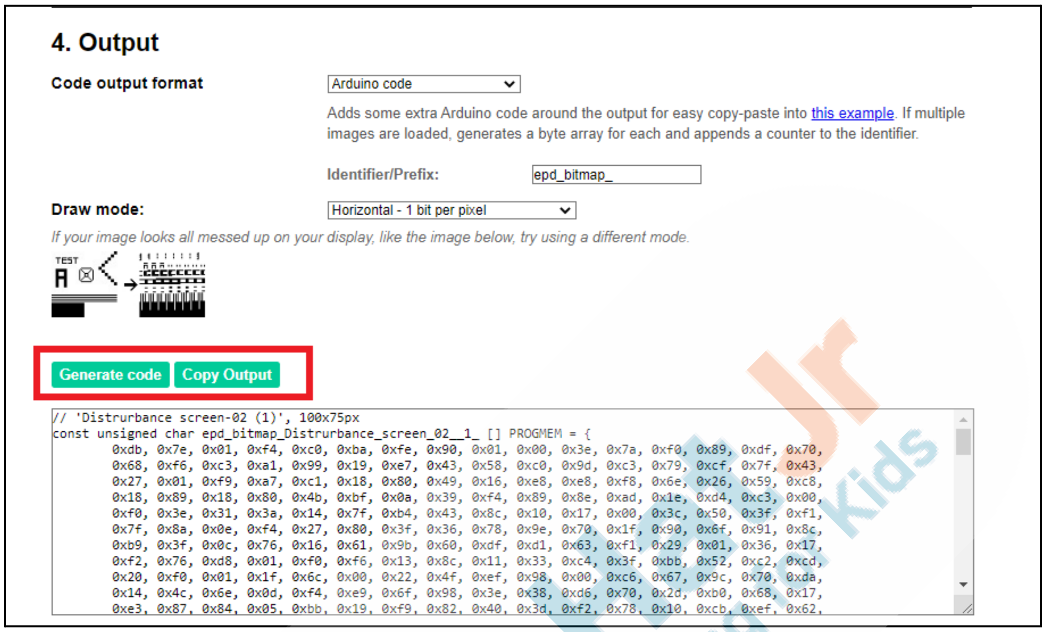
```
1 1 0 0 0 0 1 1
1 0 1 1 1 1 0 1
0 1 0 1 1 0 1 0
0 1 1 1 1 1 1 0
0 1 0 1 1 0 1 0
0 1 1 0 0 1 1 0
1 0 1 1 1 1 0 1
1 1 0 0 0 0 1 1
```

7. The bitmap of the first image has been saved in the file called **disturbance1.h**
8. Let me show you how to generate bitmap images by creating **disturbance2.h** using Teacher Activity 2.

- Click on the Choose Files button. 



- Select the image **disturbance2.png**. The PAL TV that we use in the circuit has a maximum resolution of 128x96. So images have to be resized so that they can be displayed on the TV. So if any image is larger, then you have to make it smaller to 128 x 96.
- Scroll down to the Output section and click on **Generate Code** and then **Copy Output** buttons.



## 4. Output

**Code output format** Arduino code

Adds some extra Arduino code around the output for easy copy-paste into this example. If multiple images are loaded, generates a byte array for each and appends a counter to the identifier.

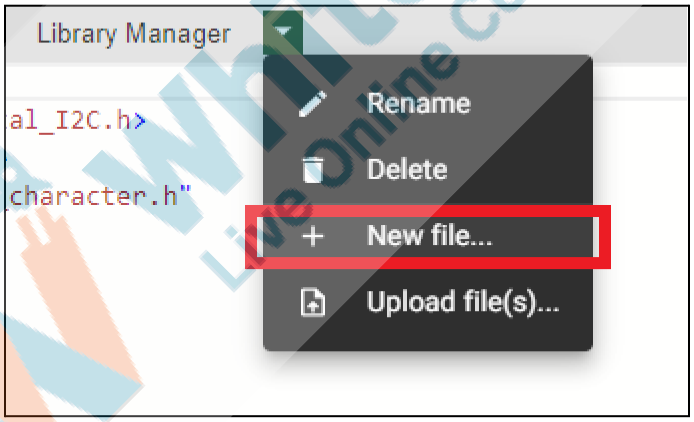
**Identifier/Prefix:** epd_bitmap_

**Draw mode:** Horizontal - 1 bit per pixel

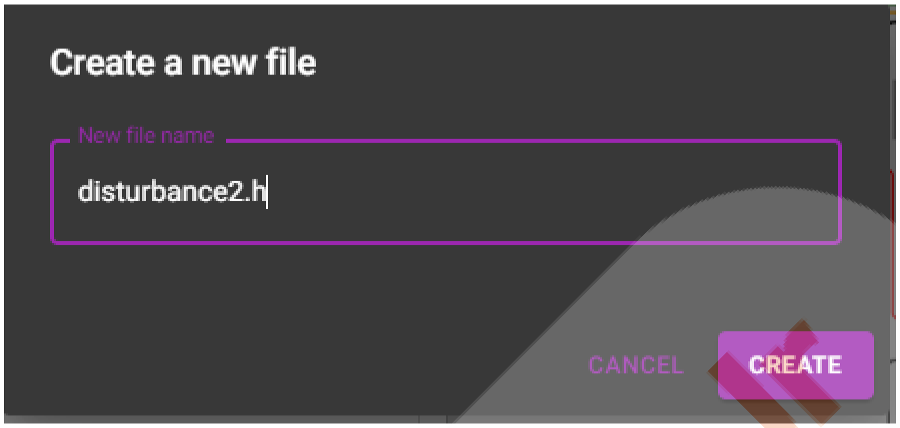
*If your image looks all messed up on your display, like the image below, try using a different mode.*

```
// 'Distrurbance screen-02 (1)', 100x75px
const unsigned char epd_bitmap_Distrurbance_screen_02__1_ [] PROGMEM = {
	0xdb, 0x7e, 0x01, 0xf4, 0xc0, 0xba, 0xfe, 0x90, 0x01, 0x00, 0x3e, 0x7a, 0xf0, 0x89, 0xdf, 0x70,
	0x68, 0xf6, 0xc3, 0xa1, 0x99, 0x19, 0xe7, 0x43, 0x58, 0xc0, 0x9d, 0xc3, 0x79, 0xcf, 0x7f, 0x43,
	0x27, 0x01, 0xf9, 0xa7, 0xc1, 0x18, 0x80, 0x49, 0x16, 0xe8, 0xe8, 0xf8, 0x6e, 0x26, 0x59, 0xc8,
	0x18, 0x89, 0x18, 0x80, 0x4b, 0xbf, 0x0a, 0x39, 0xf4, 0x89, 0x8e, 0xad, 0x1e, 0xd4, 0xc3, 0x00,
	0xf0, 0x3e, 0x31, 0x3a, 0x14, 0x7f, 0xb4, 0x43, 0x8c, 0x10, 0x17, 0x00, 0x3c, 0x50, 0x3f, 0xf1,
	0x7f, 0x8a, 0x0e, 0xf4, 0x27, 0x80, 0x3f, 0x36, 0x78, 0x9e, 0x70, 0x1f, 0x90, 0x6f, 0x91, 0x8c,
	0xb9, 0x3f, 0x0c, 0x76, 0x16, 0x61, 0x9b, 0x60, 0xdf, 0xd1, 0x63, 0xf1, 0x29, 0x01, 0x36, 0x17,
	0xf2, 0x76, 0xd8, 0x01, 0xf0, 0xf6, 0x13, 0x8c, 0x11, 0x33, 0xc4, 0x3f, 0xbb, 0x52, 0xc2, 0xcd,
	0x20, 0xf0, 0x01, 0x1f, 0x6c, 0x00, 0x22, 0x4f, 0xef, 0x98, 0x00, 0xc6, 0x67, 0x9c, 0x70, 0xda,
	0x14, 0x4c, 0x6e, 0x0d, 0xf4, 0xe9, 0x6f, 0x98, 0x3e, 0x38, 0xd6, 0x70, 0x2d, 0xb0, 0x68, 0x17,
	0xe3, 0x87, 0x84, 0x05, 0xbb, 0x19, 0xf9, 0x82, 0x40, 0x3d, 0xf2, 0x78, 0x10, 0xcb, 0xef, 0x62,
```

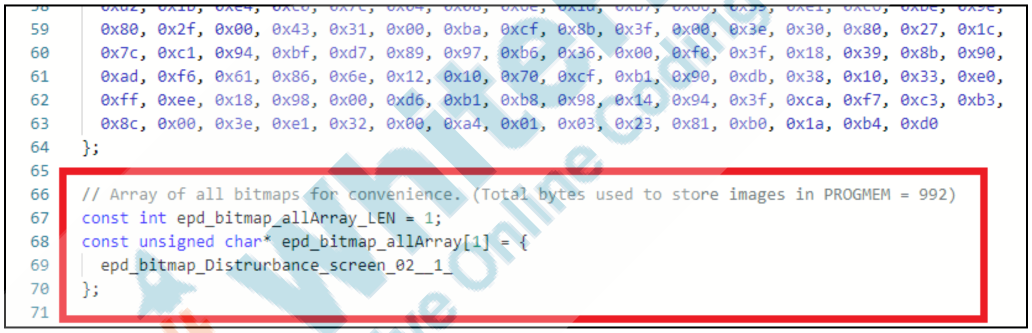
- Click on **New File** on your project and paste the bitmap.



- Name the file **disturbance2.h**



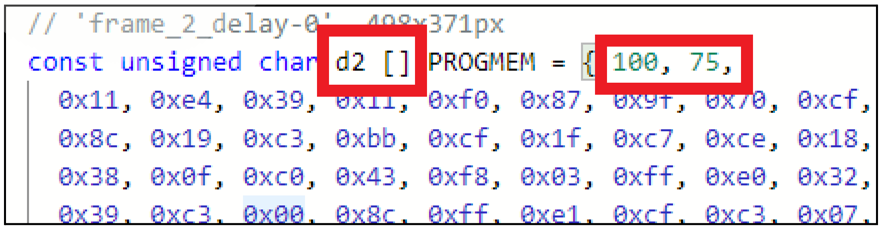
- The clipboard will have the bitmap contents. Paste it in the new file **disturbance2.h**.
- Delete the or comment off the last few lines in the file.



```
  0x80, 0x2f, 0x00, 0x43, 0x31, 0x00, 0xba, 0xcf, 0x8b, 0x3f, 0x00, 0x3e, 0x30, 0x80, 0x27, 0x1c,
  0x7c, 0xc1, 0x94, 0xbf, 0xd7, 0x89, 0x97, 0xb6, 0x36, 0x00, 0xf0, 0x3f, 0x18, 0x39, 0x8b, 0x90,
  0xad, 0xf6, 0x61, 0x86, 0x6e, 0x12, 0x10, 0x70, 0xcf, 0xb1, 0x90, 0xdb, 0x38, 0x10, 0x33, 0xe0,
  0xff, 0xee, 0x18, 0x98, 0x00, 0xd6, 0xb1, 0xb8, 0x98, 0x14, 0x94, 0x3f, 0xca, 0xf7, 0xc3, 0xb3,
  0x8c, 0x00, 0x3e, 0xe1, 0x32, 0x00, 0xa4, 0x01, 0x03, 0x23, 0x81, 0xb0, 0x1a, 0xb4, 0xd0
};

// Array of all bitmaps for convenience. (Total bytes used to store images in PROGMEM = 992)
const int epd_bitmap_allArray_LEN = 1;
const unsigned char* epd_bitmap_allArray[1] = {
  epd_bitmap_Distrurbance_screen_02__1_
};
```

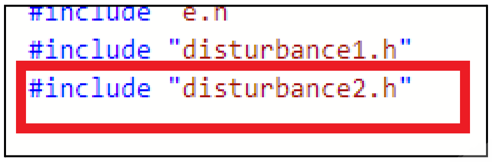
- Scroll up and rename the array to **d2** and give the proper size of the file within the curly brackets **{** .

```
const unsigned char d2[] PROGMEM = {100,75,
```



```
// 'frame_2_delay-0', 498x371px
const unsigned char d2 [] PROGMEM = { 100, 75,
  0x11, 0xe4, 0x39, 0x11, 0xf0, 0x87, 0x9f, 0x70, 0xcf,
  0x8c, 0x19, 0xc3, 0xbb, 0xcf, 0x1f, 0xc7, 0xce, 0x18,
  0x38, 0x0f, 0xc0, 0x43, 0xf8, 0x03, 0xff, 0xe0, 0x32,
  0x39, 0xc3, 0x00, 0x8c, 0xff, 0xe1, 0xcf, 0xc3, 0x07,
```

9. Add the header file **disturbance2.h**



```
#include "e.h"
#include "disturbance1.h"
#include "disturbance2.h"
```

10. Now the bitmap conversions of both images of the static video are ready in the files **disturbance1.h** and **disturbance2.h.** **disturbance1** is loaded into an array called **d1** and **disturbance2** is loaded into an array called **d2**.

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

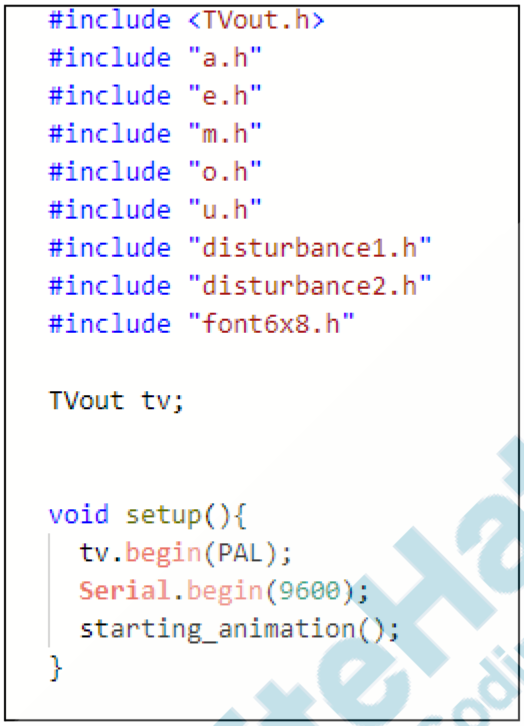
```
Serial.begin(9600);
```

12. The output to TV can be enabled by calling the **begin()** method as shown. It accepts a parameter mode, which defines whether the mode of the TV connected is PAL or NTSC. Since we have added a PAL TV we will pass the keyword PAL to the method. In case we want to change the solution of the TV, we can pass the width and height as the second and third arguments. If we do not mention them explicitly, the default resolution of 128x96 is taken.

```
tv.begin(PAL);
```

13. Let us define a method called **starting_animation()** and write the code to display the static video in this method. **starting_animation()** will be called from the **setup()** method.

Our code should look like this -



```
#include <TVout.h>
#include "a.h"
#include "e.h"
#include "m.h"
#include "o.h"
#include "u.h"
#include "disturbance1.h"
#include "disturbance2.h"
#include "font6x8.h"

TVout tv;


void setup(){
  tv.begin(PAL);
  Serial.begin(9600);
  starting_animation();
}
```

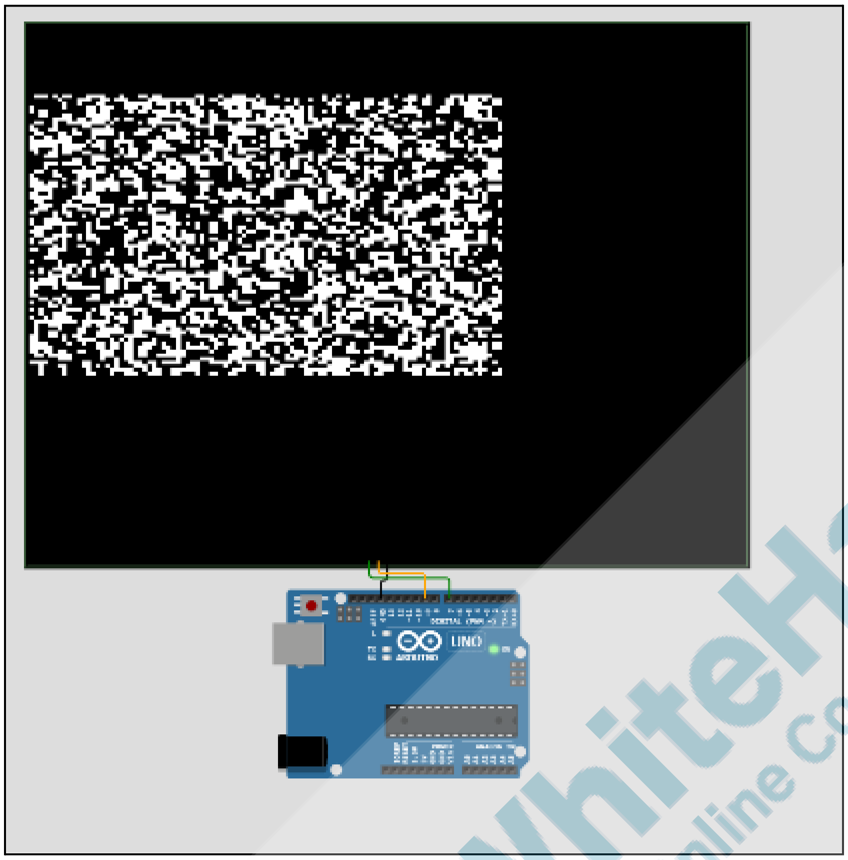
Now it's time to define the method **starting_animation()**

**tv.bitmap(x,y,array);** Is a method of the TV library that displays an image on the TV.

It accepts three parameters:

1. x coordinate to display the image
2. y coordinate to display the image
3. An array that holds the bitmap of the image.

Whenever the TV displays an image the TV library adds a border around it. An image that is displayed that is at (0,0) would have a margin on the top.



So to display the image at the center we will use x coordinate as 20 and y coordinate as 16.

**tv.bitmap(20,16, d1)** displays the d1 image at x-coordinate 20 and y coordinate 16.

```
tv.bitmap(20,16,d1);
```

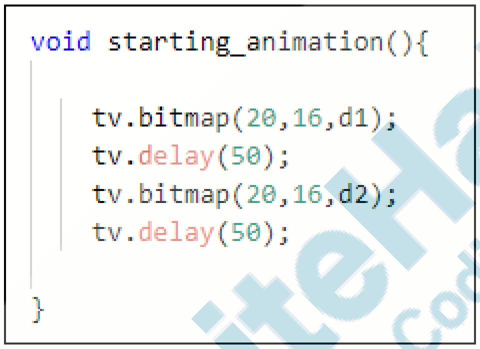
Now after a small delay of 50 milliseconds, let us display the next image stored in the array d2.

```
tv.delay(50);
tv.bitmap(20,16,d2);
```

The code snippet could be written as:

- Display an image
- Add a delay
- Display another image
- Add a delay

The code would look like this



```
void starting_animation(){

    tv.bitmap(20,16,d1);
    tv.delay(50);
    tv.bitmap(20,16,d2);
    tv.delay(50);

}
```

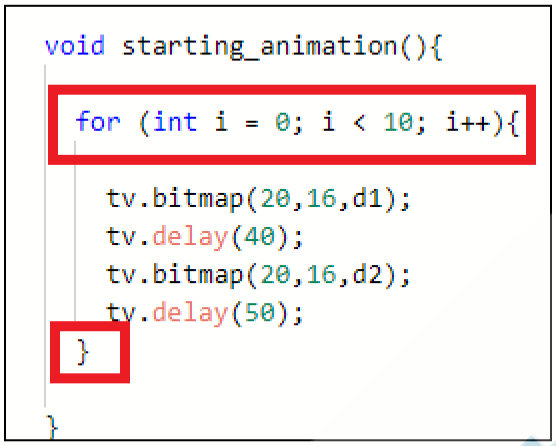
But the images are displayed for a very short amount of time.

What should we do if we want to repeat the same lines of code multiple times?

You are right, so let us continuously display the images by using a **for loop.**

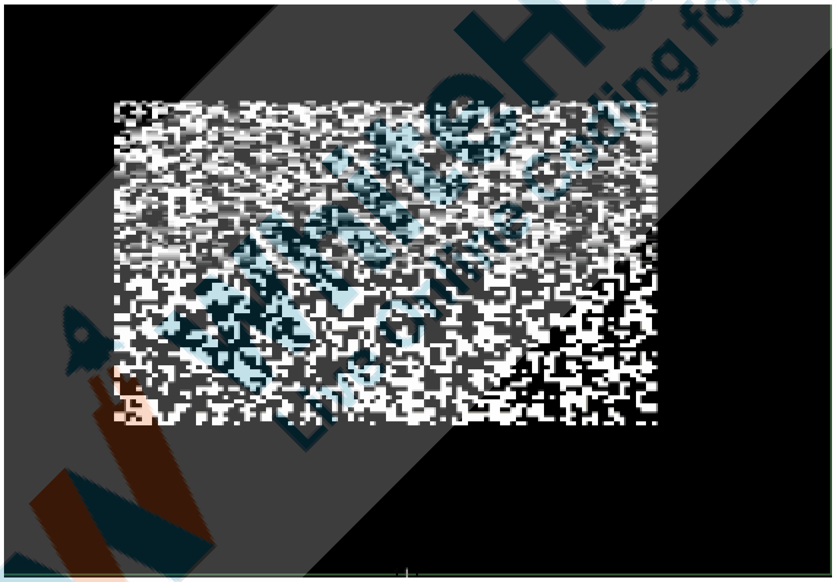
**ESR:** Write the piece of code in a loop.

**CODE:**



```
void starting_animation(){
  for (int i = 0; i < 10; i++){
    tv.bitmap(20,16,d1);
    tv.delay(40);
    tv.bitmap(20,16,d2);
    tv.delay(50);
  }
}
```

**OUTPUT:**



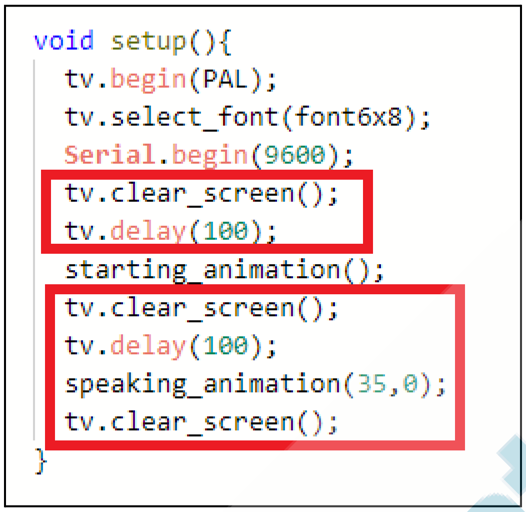
Isn't that great! We successfully displayed the first static video.

**Teacher Stops Screen Share**

| | |
|---|---|
| Now it's your time to code. Let us display images which speak out the letters ‘a’, ‘e’,‘m’, ‘o’, ‘u’. | |

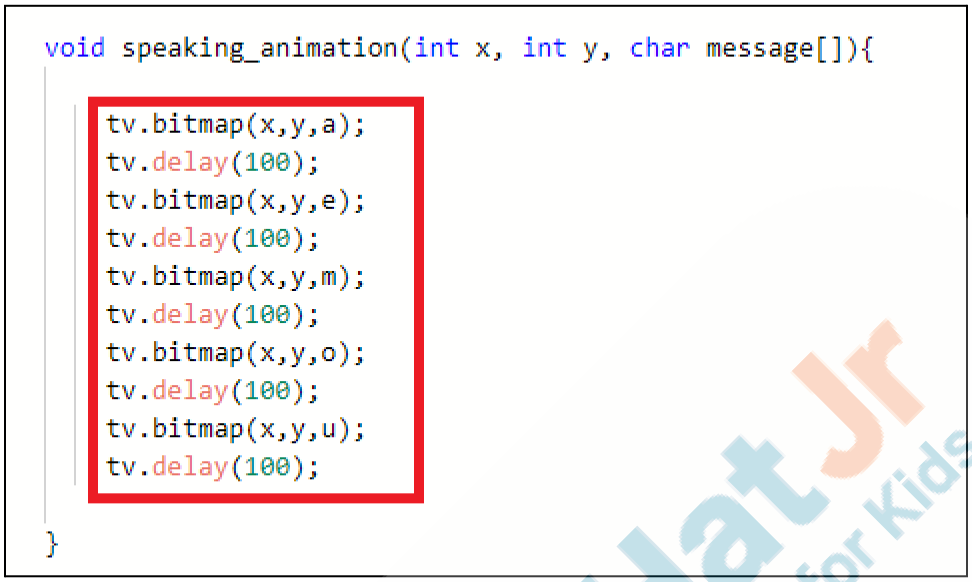
| | |
|---|---|
| Can you display the speaking images ?<br><br>Let's try. I will guide you through it. | **ESR:** Yes, sure! |
| **STUDENT-LED ACTIVITY- 15 mins** | |
| • **Ask the student to press the ESC key to come back to the panel.**<br>• **Guide the student to start Screen Share.**<br>• **The teacher gets into Full Screen.** | |
| **Student Initiates Screen Share** | |
| ACTIVITY<br>• **Define a method called speaking_animation() to display the bitmap images of speaking faces.**<br>• **Learn to display text along with images.** | |
| Teacher Action | Student Action |
| *Teacher guides the student to download boilerPlate code from Student Activity 2.* | Student opens wokwi simulator. |
| *Guide the students to delete the existing files in the editor. Unzip the downloaded file and upload it in the wokwi project folder.* | |
| The connections are already done and complete.<br><br>When you click on the play button, what do you observe?<br><br>Do you remember what your task for the day is? | **ESR:** The static video is displayed.<br><br>**ESR:** The bitmap of 5 images representing speaking images of ‘a’, ‘e’,‘m’, ‘o’, ‘u’ |

| | |
|---|---|
| Exactly, you have to display the speaking animation on the PAL TV.<br><br>But before we start displaying them there is an important task to be done. Can you guess what it is?<br><br>We have to clear the TV screen so that there are no overlapping images.<br>There is a method named **clear_screen()** that can be called to clear the screen.<br><br>`tv.clear_screen();`<br><br>We will clear the screen three times:<br>1. Before any display starts<br>2. After starting animation<br>3. After speaking images<br><br>Everytime we clear the screen, we will pause for 100 milliseconds so that it stays clear for some time.<br><br>Displaying the speaking images can be written in a method called **speaking_animation()**.<br><br>Go ahead and write the code inside the **setup()** method. | **ESR:** Varied! |
| **CODE:** | |



```
void setup(){
  tv.begin(PAL);
  tv.select_font(font6x8);
  Serial.begin(9600);
  tv.clear_screen();
  tv.delay(100);
  starting_animation();
  tv.clear_screen();
  tv.delay(100);
  speaking_animation(35,0);
  tv.clear_screen();
}
```

| | |
|---|---|
| Now let us define the **speaking_animation()** method.<br><br>The bitmaps of the 5 images representing 'a', 'e','m', 'o', 'u' are given in the boilerplate.<br><br>We can use a similar approach that we used to display the disturbance or static images.<br><br>Do you remember which method we had used to display the bitmap of images?<br><br>Yes, the method tv.bitmap() can be used.<br><br>**tv.bitmap(x,y,array_name)**<br><br>Let us call the method for each of the images one by one. Adding a delay of 100 milliseconds will bring a pause so that the image can be seen on the screen for a while. | **ESR:**. The method **tv.bitmap()** was used. |
| **CODE:** | |



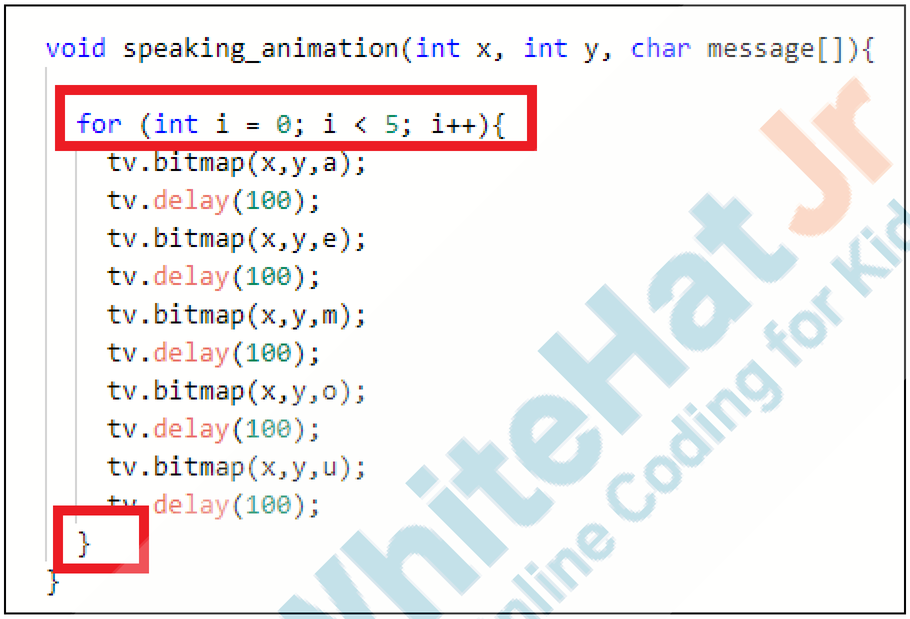
```
void speaking_animation(int x, int y, char message[]){

    tv.bitmap(x,y,a);
    tv.delay(100);
    tv.bitmap(x,y,e);
    tv.delay(100);
    tv.bitmap(x,y,m);
    tv.delay(100);
    tv.bitmap(x,y,o);
    tv.delay(100);
    tv.bitmap(x,y,u);
    tv.delay(100);

}
```

**OUTPUT:**

| | |
|---|---|
| Great, The video was long enough to speak out a word.<br><br>What can we do so that it looks like a full sentence is spoken out? | **ESR:** Display the images multiple times. |

| | |
|---|---|
| Yes, you have said it right ! Let us use a for loop so that the images are displayed multiple times. | |

**CODE:**

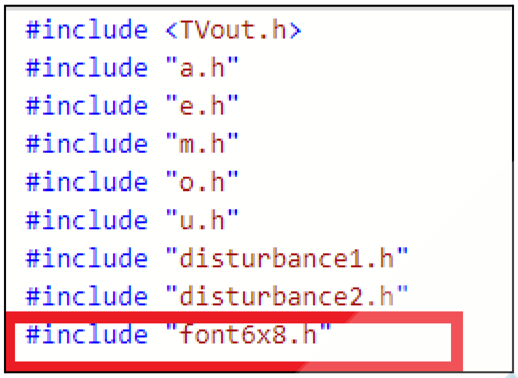


```
void speaking_animation(int x, int y, char message[]){
  for (int i = 0; i < 5; i++){
    tv.bitmap(x,y,a);
    tv.delay(100);
    tv.bitmap(x,y,e);
    tv.delay(100);
    tv.bitmap(x,y,m);
    tv.delay(100);
    tv.bitmap(x,y,o);
    tv.delay(100);
    tv.bitmap(x,y,u);
    tv.delay(100);
  }
}
```

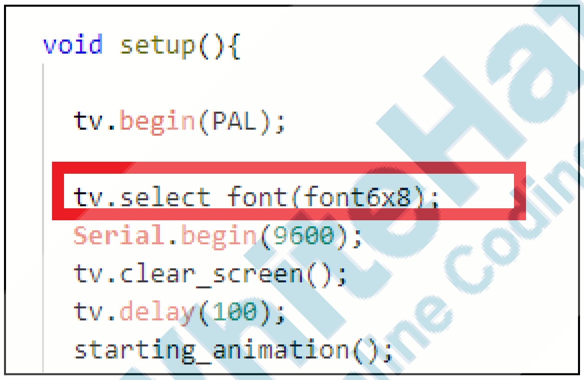
**OUTPUT:**

| | |
|---|---|
| Have you noticed that most news channels will have their name written or some news written at the bottom along with the visuals?<br><br>Let us learn how to display text on a PAL TV.<br>The **TVout** library has a method, print**()** that prints a character array to the output.<br><br>It accepts three parameters :<br>1. x coordinate to display the message<br>2. y coordinate to display the message<br>3. Message to be displayed: the message has to be passed as a character array.<br><br>We can change the font of the display by adding the preferred ont file in the header<br><br>`#include "font6x8.h"`<br><br>and selecting the font before displaying the text.<br><br>`tv.select_font(font6x8);` | **ESR:** Yes! |

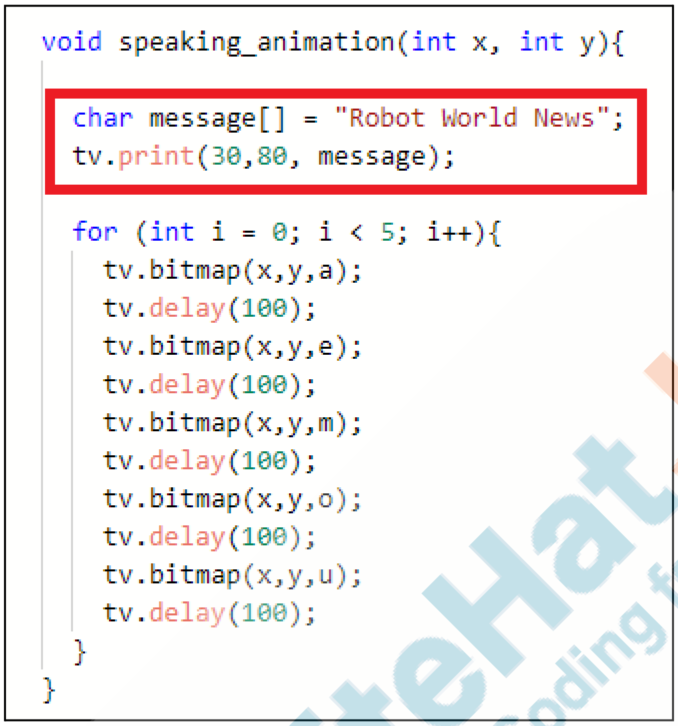
**Reference Code:**



```
#include <TVout.h>
#include "a.h"
#include "e.h"
#include "m.h"
#include "o.h"
#include "u.h"
#include "disturbance1.h"
#include "disturbance2.h"
#include "font6x8.h"
```

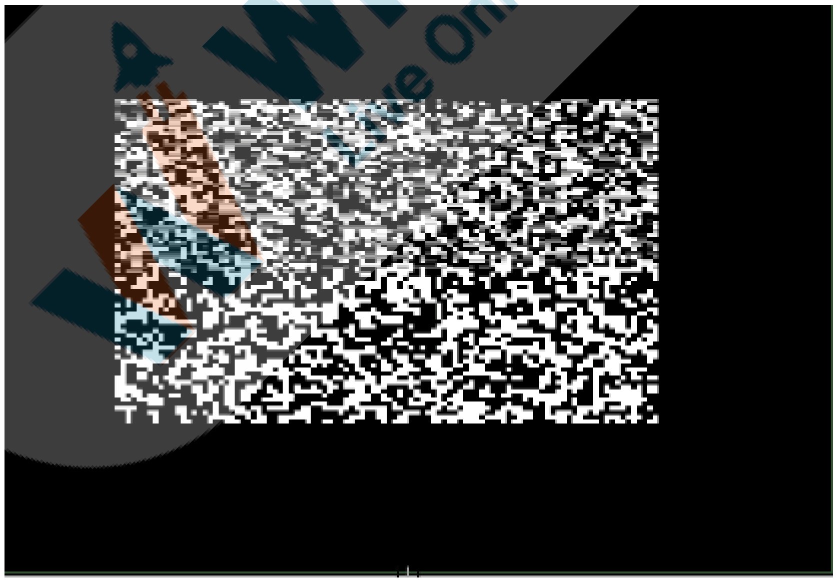


```
void setup(){

  tv.begin(PAL);

  tv.select_font(font6x8);
  Serial.begin(9600);
  tv.clear_screen();
  tv.delay(100);
  starting_animation();
```



```
void speaking_animation(int x, int y){

  char message[] = "Robot World News";
  tv.print(30,80, message);

  for (int i = 0; i < 5; i++){
    tv.bitmap(x,y,a);
    tv.delay(100);
    tv.bitmap(x,y,e);
    tv.delay(100);
    tv.bitmap(x,y,m);
    tv.delay(100);
    tv.bitmap(x,y,o);
    tv.delay(100);
    tv.bitmap(x,y,u);
    tv.delay(100);
  }
}
```

**Reference Output:**



Click here to view the reference video.

**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

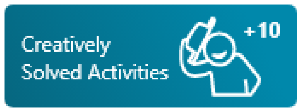
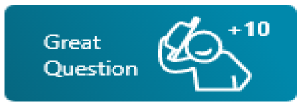
**Activity Details**

**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!<br><br>In the next class, we will learn how to add a remote and television together on the same circuit. | *Make sure you have given at least 2 hats-off during the class for:*<br> <br> |

| | |
|---|---|
| | Strong Concentration +10 |

**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 | image2cpp | https://javl.github.io/image2cpp/ |
| Teacher Reference 1 | PAL documentation | https://docs.wokwi.com/parts/wokwi-ir-receiver |
| Teacher Reference 2 | Boilerplate Code | https://github.com/procodingclass/PRO-C276-Teacher-Boilerplate |
| Teacher Reference 3 | Reference Code | https://github.com/procodingclass/PRO-C276-Reference-Code |
| Teacher Reference 4 | Project | https://s3-whjr-curriculum-uploads.whjr.online/99a0155a-d6b3-4570-9c31-52bbeee14c92.pdf |
| Teacher Reference 5 | Project Solution | https://github.com/procodingclass/PRO-C276-Project-Solution |

| | | |
|---|---|---|
| Teacher Reference 6 | In-Class Quiz | https://s3-whjr-curriculum-uploads.whjr.online/8e25535c-133e-4518-a986-548543e7c572.pdf |
| Student Activity 1 | Simulator | https://wokwi.com/ |
| Student Activity 2 | Student Activity Boilerplate code | https://github.com/procodingclass/PRO-C276-Student-Boilerplate |
| Student Activity 3 | PAL documentation | https://docs.wokwi.com/parts/wokwi-ir-receiver |