

PERSONAL TELEVISION - 1



What is our GOAL for this CLASS?

In this class, we learned about PAL TV and connected it to an Arduino UNO board. You learned to display text and images on the TV.

What did we ACHIEVE in the class TODAY?

- Understand about PAL TV.
- Learn to connect a TV to an Arduino board.
- Understand the basics of bitmap.
- Learn to display images using bitmap.
- Learn to display text on the TV.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Concepts : PAL TV, analog and digital signals, bitmap
- Coding blocks : TV-out.h, tv.bitmap(), tv.clear_screen(), tv.delay().

How did we DO the activities?

1. Working principle of PAL TV:

The TV component that we are connecting to the circuit is called **PAL TV**. PAL which is short form for **Phase Alternating Line (PAL)** is a color encoding system for television. A PAL picture is made up of 625 interlaced lines and is displayed at a rate of 25 frames per second.

2. Create the circuit:

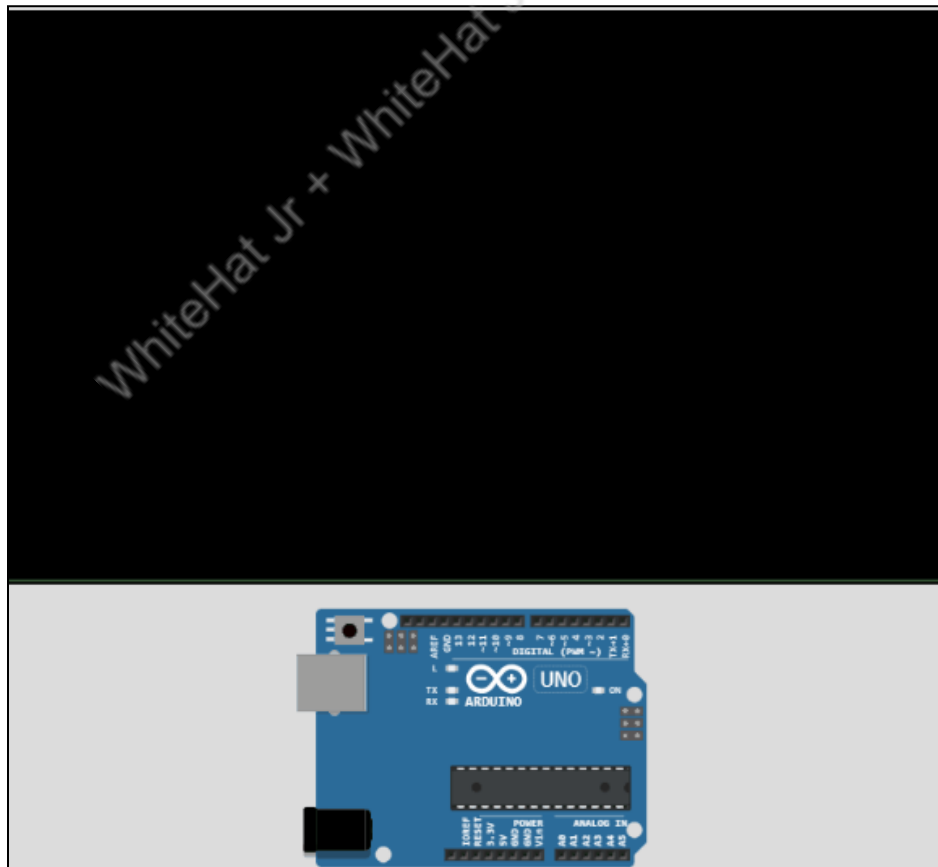
a. Components:

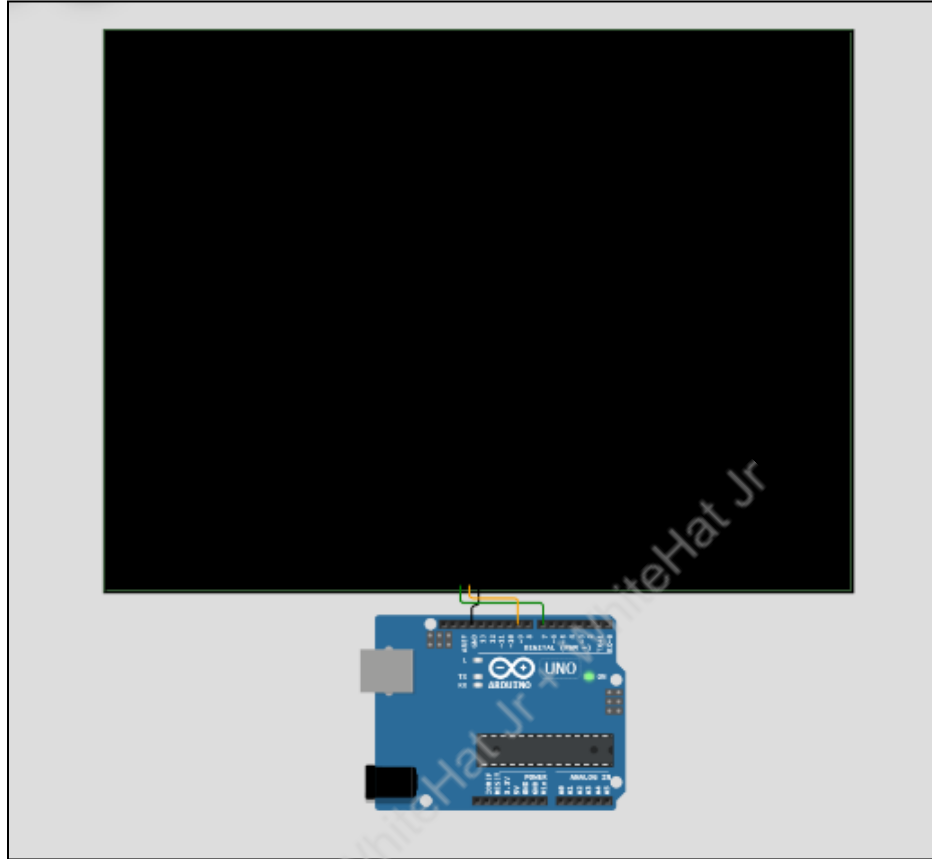
- 1 x Arduino Uno board
- 1 x PAL TV

b. Connections:

The circuit of this project consists of an Arduino Uno board, and **PAL TV**. Connect the Arduino Uno board and **PAL TV** component. 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.

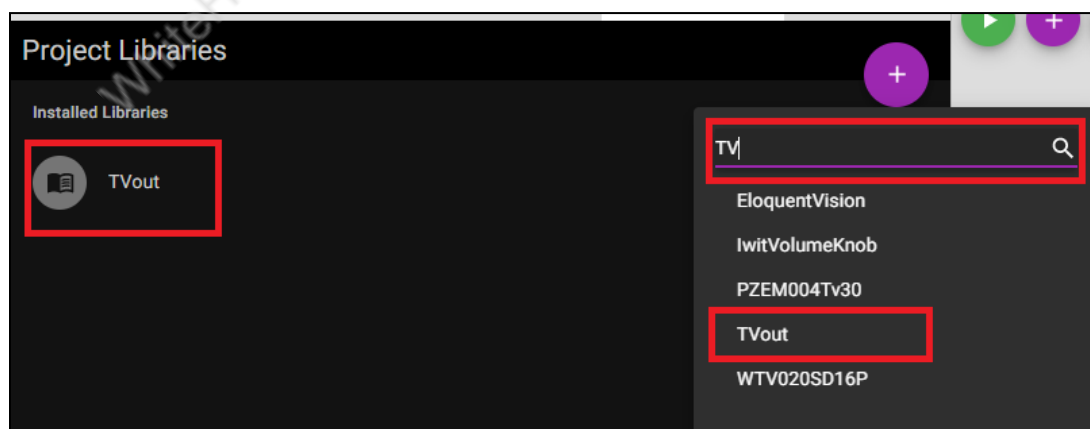
PAL TV	Arduino Uno PIN
GND	GND
VIDEO	7
SYNC	9





3. Code:

- a. First, go to the **Library Manager** and **Add a new library** called **TVout**.



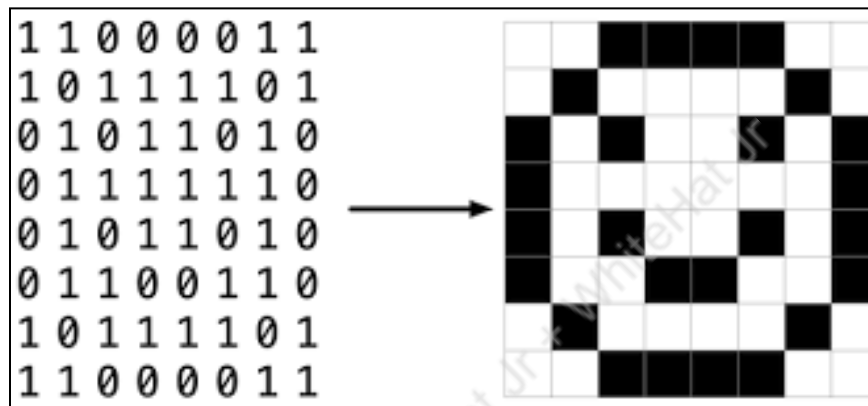
- b. Now, go to **sketch.ino**, add the header file.

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

- c. Let's make an object of the TVout class.

```
TVout tv;
```

- d. PAL TV uses bitmaps of images to display them. Bitmaps use 0s and 1s to represent an image.



- e. We can convert any image to its corresponding bitmap.

- Click on the Choose Files button.

Choose Files No file chosen

1. Select image

Choose Files No file chosen

or

1. Paste byte array

128 x 64 px

Read as horizontal Read as vertical

- Select the image **disturbance2.png**.
- Scroll down to the Output section and click on **Generate Code**

Generate code

 and then **Copy Output**

Copy Output

 buttons.

4. Output

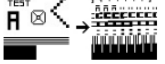
Code output format Arduino code

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

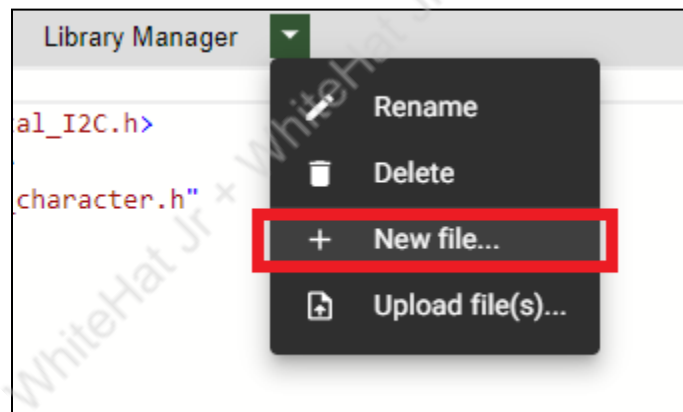
If your image looks all messed up on your display, like the image below, try using a different mode.



Generate code Copy Output

```
// 'Disturbance screen-02 (1)', 100x75px
const unsigned char epd_bitmap_Disturbance_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, 0xc7, 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, 0xb5, 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**

Create a new file

New file name

CANCEL CREATE

- The clipboard will have the bitmap contents. Paste it in the new file

disturbance.h.

- Delete the or comment off the last few lines in the file

```

58 0x02, 0x10, 0xe4, 0xc0, 0x7c, 0x04, 0xb0, 0x0e, 0x10, 0x07, 0x00, 0x33, 0xe1, 0xc0, 0x0e, 0x3e,
59 0x80, 0x2f, 0x00, 0x43, 0x31, 0x00, 0xba, 0xcf, 0x8b, 0x3f, 0x00, 0x3e, 0x30, 0x80, 0x27, 0x1c,
60 0x7c, 0xc1, 0x94, 0xbf, 0xd7, 0x89, 0x97, 0xb6, 0x36, 0x00, 0xf0, 0x3f, 0x18, 0x39, 0x8b, 0x90,
61 0xad, 0xf6, 0x61, 0x86, 0x6e, 0x12, 0x10, 0x70, 0xcf, 0xb1, 0x90, 0xdb, 0x38, 0x10, 0x33, 0xe0,
62 0xff, 0xee, 0x18, 0x98, 0x00, 0xd6, 0xb1, 0xb8, 0x98, 0x14, 0x94, 0x3f, 0xca, 0xf7, 0xc3, 0xb3,
63 0x8c, 0x00, 0x3e, 0xe1, 0x32, 0x00, 0xa4, 0x01, 0x03, 0x23, 0x81, 0xb0, 0x1a, 0xb4, 0xd0
64 };
65
66 // Array of all bitmaps for convenience. (Total bytes used to store images in PROGMEM = 992)
67 const int epd_bitmap_allArray_LEN = 1;
68 const unsigned char* epd_bitmap_allArray[1] = {
69   epd_bitmap_Disturbance_screen_02__1_
70 };
71

```

- 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,.....`

```

// 'Disturbance screen-02', 100x75px
const unsigned char d2[] PROGMEM = {100,75,
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, 0xa0, 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,

```

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

```

#include <e.h>
#include "disturbance1.h"
#include "disturbance2.h"

```

f. Now, let's write the following code inside the **setup()** function.

- Initialize communication

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

- The output to TV can be enabled by calling the **begin()** method as shown

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

- Include the header files of all the images and font6x8.h for the preferred font.

The code should look like this now -

```
#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();
}
```

g. Now define the method **starting_animation()**

- Use **tv.bitmap()** to display the disturbance1 and disturbance2 images.
tv.bitmap() accepts three parameters.
 - x coordinate to display the image
 - y coordinate to display the image
 - 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.



- disturbance1 is stored in array d1 and disturbance2 is stored in array d2. Use a for loop to display the images for a longer duration of time. To display the image at the center use x coordinate as 20 and y coordinate as 16.

CODE:

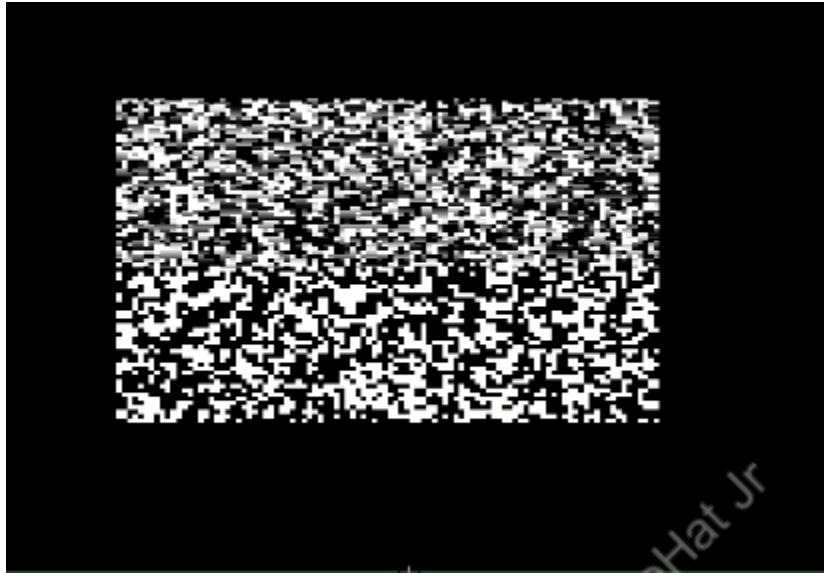
```
void starting_animation(){  
    for (int i = 0; i < 10; i++){  
        tv.bitmap(0,0,d1);  
        tv.delay(40);  
        tv.bitmap(0,0,d2);  
        tv.delay(50);  
    }  
}
```

OUTPUT:

© 2021 - WhiteHat Education Technology Private Limited.

Note: This document is the original copyright of WhiteHat Education Technology Private Limited.

Please don't share, download or copy this file without permission.



h. Now define the method `speaking_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);  
    }  
}
```

i. `tv.select_font ()` sets the font

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

j. We will clear the screen three times so that the images are spaced evenly and looks like a video:

- Before any display starts
- After starting animation
- After speaking images

```
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();  
}
```

[Click here](#) to view the reference output video.

What's NEXT?

In the **next class**, we will learn to connect a remote and TV to an Arduino board.

Expand Your Knowledge

To know more about **PAL TV** on wokwi, [click here](#).