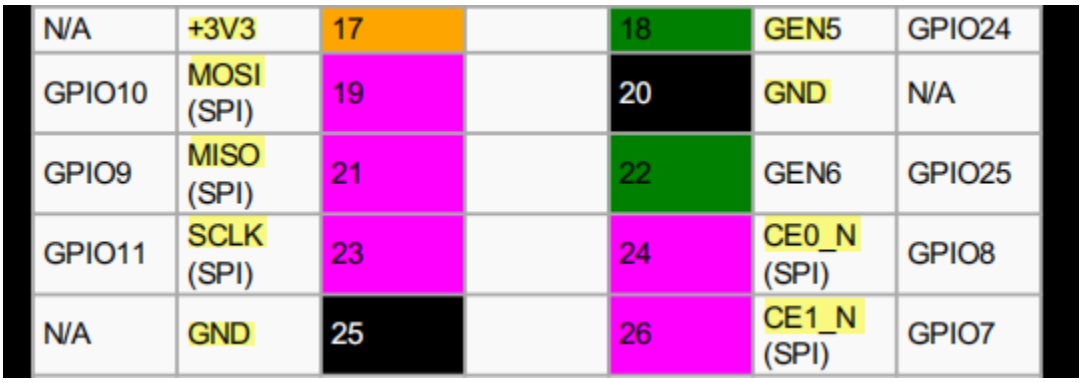
1. The pipe has no name, it is generated for one use and both ends must be inherited from the single process which created the pipe. The FIFO file is almost similar with the pipe but instead of being an anonymous, temporary connection, it uses a name or names like any other file. FIFO is opened by name by the processes to achieve the communication through it.

2. The guide is making the FIFO to control the mplayer, but the file name is named as "mplayer", thus it is not like a FIFO file, it is a mplayer file to control. Thus, the naming is a little confusing.

3. The display and touchscreen use the hardware SPI pins (SCK, MOSI, MISO, CE0, CE1) as well as GPIO #25 and #24. And the 3.3V power supply and the ground is also connected to the piTFT at the pin 17 and 20 respectively.



N/A	+3V3	17		18	GEN5	GPIO24
GPIO10	MOSI (SPI)	19		20	GND	N/A
GPIO9	MISO (SPI)	21		22	GEN6	GPIO25
GPIO11	SCLK (SPI)	23		24	CE0_N (SPI)	GPIO8
N/A	GND	25		26	CE1_N (SPI)	GPIO7

As shown in the picture above, the pin used for piTFT is some GPIO pins to achieve SPI and GEN and also the GPIO pins used for the four buttons. The following table will list the pins used by piTFT screen and their functions.

Pin number	Function	GPIO number
pin19	MOSI(SPI)	GPIO10
pin21	MISO(SPI)	GPIO9
pin23	SCLK(SPI)	GPIO11
pin24	CE0_N	GPIO8
pin25	CE1_N	GPIO7
pin18	GEN5	GPIO24
pin22	GEN6	GPIO25
pin11	GEN0	GPIO17
pin15	GEN3	GPIO22
pin16	GEN4	GPIO23
pin13	GEN3	GPIO27

4. According to the pin diagram, all the GPIO pins could be used for general purpose. Thus, the maximum set when not using special functions are 28.

Because all the GPIO pins could be used for special functions, hence the minimum set is 0.

5. For the “start_video” shell script, the command for running the python file “video_control.py” is ended with a “&” to let it run in the background and it was placed at the first to run, while the video running command was placed at the second and in the foreground side.

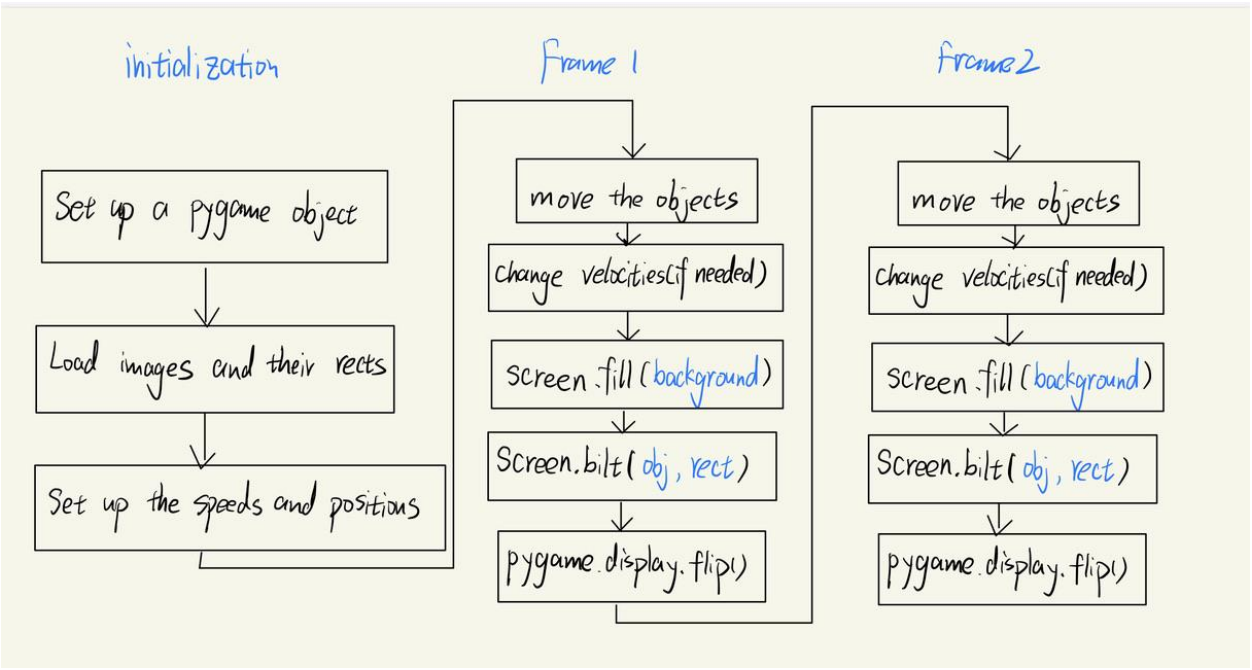
6. (According to https://raspberrypi.stackexchange.com/questions/4370/where-does-the-raspberry-pi-get-the-time-from) RPi uses the Network Time Protocol (NTP) Server to get the accurate time based on the time zone we chose. So as long as we have the Internet connection, we can synchronize the right time with the server. We may need to use an external, battery-backed real-time clock when we cannot connect the RPi to the Internet.

7. If we do not use the R2 and press the button, the current will be 3.3V/0, which will cost a damage to the GPIO. Although with software we can use GPIO.setup() to deal with it, the GPIO has no protection before the running of the code. On the other hand, after we installing the R2, the current will become 3.3mA, which is the less than 50mA, the maximum value for GPIO pins.

8. In this case, when running the python without a fifo created, when pressing buttons, the python program will write commands to the non-existed video_fifo. When the system detects the target file is not existed, it will create a new file named 'video_fifo', but it's not a fifo file but an ordinary text file has the same name, so the file will not be created correctly. The running video then will get commands from this text file so it actually receives no command, and we cannot control the video with the python program.

9. **a)** Possibility 1: The button is cheap is when we remove our finger to 'un-press', the inside circuits may haven't separated yet and still stick together so the program detects that the button is still pressed, resulting in triggering the same command multiple times.
   b) Possibility 2: If we do not use the call back but the while version, in the while loop, we set up a "time.sleep(T)" statements. If we keep pressing the button for more than time T, the program will detect the button pressed more than once, triggering the same command.

10. When we develop a pygame game, we first create screen object with 'pygame.display.set_mode()' which also sets up the screen size. If we want to animate an image, first we need to load the image with "pygame.image.load()", when we get the rectangle object of the image using "get_rect()". We need to initialize the starting coordinate and speed of the image. In the mail function, we use a forever loop to keep updating the game screen by:
    1) calculate the current position of the image using rect.move();
    2) fill the screen with a black background using "fill()";
    3) add images to the screen using "bilt(<image_object>, <rect>);
    4) update the showing screen with "pygame.display.flip()";
    Based on these steps, the diagram is:


initialization
Set up a pygame object
Load images and their rects
Set up the speeds and positions
Frame 1
move the objects
change velocities(if needed)
screen.fill(background)
Screen.bilt(obj, rect)
pygame.display.flip()
Frame 2
move the objects
change velocities(if needed)
screen.fill(background)
Screen.bilt(obj, rect)
pygame.display.flip()