**6.**

(According to [https://raspberrypi.stackexchange.com/questions/4370/where-does-the- raspberry-pi-get-the-time-from](https://raspberrypi.stackexchange.com/questions/4370/where-does-the-%20raspberry-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.

**c)** Possibility 3:

**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:

