main.py

```
import pygame
import sys
import spidev
import time
import re
import argparse
import socket
from sh import omxplayer, aplay
import RPi.GPIO as GPIO

# parse arguments
parser = argparse.ArgumentParser()
parser.add_argument("name", help="name of the diary")
parser.add_argument("address", help="ip address of server")
parser.add_argument("port", help="port of server", type=int)
args = parser.parse_args()

# media files
VIDEO_NAME = args.name + ".mp4"
AUDIO_NAME = args.name + ".wav"
IMAGE_NAME = args.name + ".jpg"

# server address
ADDR = args.address
PORT = args.port

# screen
IMAGE_WIDTH = 1024
IMAGE_HEIGHT = 768

# buttons input
GPIO_SENSORS = [4, 25, 24]

# rear sensor
R_SENSOR_ID = 1
# value when near
#R_SENSOR_MIN = 700
R_SENSOR_MIN = 24
# value when far
#R_SENSOR_MAX = 300
R_SENSOR_MAX = 36

VIDEO_STARTED = False
VIDEO_PAUSED = False

# number of samples for filtering
NUM_CAMP = 5

omx_stdin = None
omx_process = None
tot_sec = None

# print splash screen
def print_intro(name):
  print("""        __...--~~~~~-._   _.-~~~~~--...__
      //               `V'               \\\
     //                 |                 \\\
    //__...--~~~~~~-._  |  _.-~~~~~~--...__\\\
   //__.....----~~~~._\ | /_.~~~~----.....__\\\
  ====================\\\|//====================
                      `---`""")
  print("\t[[[[ Diario: " + name.upper() + " ]]]]\n")


def sendUDP(value):
  sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
#  print('sendUDP ', value)
  sock.sendto(value, (ADDR, PORT))

def setup_buttons():
  GPIO.setmode(GPIO.BCM)
  [GPIO.setup(x, GPIO.IN) for x in GPIO_SENSORS]

def read_buttons(): # get index of button pressed
  button = [x for x in GPIO_SENSORS if not(GPIO.input(x))]
  return 0 if not button else GPIO_SENSORS.index(button[0])+1
```

```
# read SPI data from MCP3008 chip, 8 possible adc's (0 thru 7)
def readadc(adcnum):
  if ((adcnum > 7) or (adcnum < 0)):
    return -1
  r = spi.xfer2([1,(8+adcnum)<<4,0])
  adcout = ((r[1]&3) << 8) + r[2]
  return adcout

# interact with omxplayer STDIN and STDOUT
def interact(line, stdin, process):
  global tot_sec
  global VIDEO_PAUSED
  global omx_stdin
  global omx_process

  omx_stdin = stdin
  omx_process = process

  # video regexp
  video_curr_rexp = re.compile(r'V :\s*([\d.]+).*')
  video_total_rexp = re.compile(r'Length : *([\d.]+)*')

  # get current video time
  curr = video_curr_rexp.search(line)

  if curr and tot_sec:
    pts = curr.group(1)
    sec = int(pts.split(".")[0]) / 1000000
    print(sec, tot_sec)
    # stop video to last seconds
    if tot_sec == sec and VIDEO_PAUSED == False:
      VIDEO_PAUSED = True
      stdin.put('p')
      print("---- PAUSE ----")

  else:
    len = video_total_rexp.search(line)

    if len:
      tot_pts = len.group(1)
      tot_sec = (int(tot_pts) / 1000) - 11
      #print(tot_sec)
      # stops 2 seconds before end

print_intro(args.name)

# open communication with distance sensor
spi = spidev.SpiDev()
spi.open(0,0)

setup_buttons()

pygame.init()
pygame.mouse.set_visible(False)

# init rendering screen
displaymode = (IMAGE_WIDTH , IMAGE_HEIGHT)
screen = pygame.display.set_mode(displaymode)

# load cover image
cover = pygame.image.load(IMAGE_NAME).convert()

# set cover position
position = pygame.Rect((0, -IMAGE_HEIGHT, IMAGE_WIDTH, IMAGE_HEIGHT))
screen.blit(cover, position)

i = 0
r_acc = 0

while True:
  try:
    # read rear sensor value
    # r_sensor_value = readadc(R_SENSOR_ID)
    r_val = readadc(R_SENSOR_ID)
    r_sensor_value = 4800.0/(r_val - 18)
    if i == NUM_CAMP:

      if VIDEO_STARTED == True and omx_stdin:
        chapter = read_buttons()

        if chapter > 0:
          if VIDEO_PAUSED == True:
            omx_stdin.put('p')
            print("--- PLAY ---")
            time.sleep(.3)

          run_audio.kill()
          omx_stdin.put("\027[B")
          print('Rewind')
          time.sleep(1)

          if chapter > 1:
            print('sendOMX ' + repr(chapter))
            omx_stdin.put(repr(chapter))
          else:
            run_audio = aplay(AUDIO_NAME, _bg=True)

          VIDEO_PAUSED = False


      r_sensor_value = r_acc / NUM_CAMP

      i = 0
      r_acc = 0

      #print('R: ', r_sensor_value)

      if r_sensor_value >= R_SENSOR_MAX:

        if VIDEO_STARTED == False:
          print("Starting video...")
          sendUDP('1') # open
          VIDEO_STARTED = True
          VIDEO_PAUSED = False
          run = omxplayer('-s', VIDEO_NAME, _bg=True, _out=interact, _out_bufsize=250)
          time.sleep(1)
          run_audio = aplay(AUDIO_NAME, _bg=True)
          screen.fill((0,0,0))
          pygame.display.update()
          R_SENSOR_MAX = R_SENSOR_MAX - 3

      elif r_sensor_value <= R_SENSOR_MIN:
        sendUDP('0') # closed

      else:
        sendUDP('2') # motion

        if VIDEO_STARTED == True:
          VIDEO_STARTED = False
          R_SENSOR_MAX = R_SENSOR_MAX + 3
          print("Stopping video...")
          omx_stdin = None
          omx_process.kill()
          omx_process.wait()
          run_audio.kill()

        else:
          # redraw background
          screen.fill((0,0,0))
          new_value = ((R_SENSOR_MAX - r_sensor_value) * (-IMAGE_HEIGHT))/(R_SENSOR_MAX-
R_SENSOR_MIN)
          cover_position = pygame.Rect(0, new_value, IMAGE_WIDTH, IMAGE_HEIGHT)
          screen.blit(cover, cover_position)
          pygame.display.update()

    else:
      i += 1
      r_acc += r_sensor_value

    pygame.time.delay(100/NUM_CAMP)

  except KeyboardInterrupt:
```

```
if VIDEO_STARTED == True:
  omx_process.kill()
  omx_process.wait()
  run_audio.kill()
pygame.quit()
GPIO.cleanup()
#time.sleep(1)
sys.exit()
```

main_audio.py

```
import sys
import time
import spidev
import argparse
import socket
from sh import aplay

# parse arguments
parser = argparse.ArgumentParser()
parser.add_argument("name", help="name of the diary")
parser.add_argument("address", help="ip address of server")
parser.add_argument("port", help="port of server", type=int)
args = parser.parse_args()

AUDIO_NAME = args.name + ".wav"

ADDR = args.address
PORT = args.port

# rear sensor
R_SENSOR_ID = 1
# value when near
R_SENSOR_MIN = 7
# value when far
R_SENSOR_MAX = 28

AUDIO_STARTED = False

NUM_CAMP = 5

# read SPI data from MCP3008 chip, 8 possible adc's (0 thru 7)
def readadc(adcnum):
  if ((adcnum > 7) or (adcnum < 0)):
    return -1
  r = spi.xfer2([1,(8+adcnum)<<4,0])
  adcout = ((r[1]&3) << 8) + r[2]
  val = (int)(4800/(adcout - 20))
  return val

def sendUDP(value):
  sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
  #print('sendUDP ', value)
  sock.sendto(value, (ADDR, PORT))

# open communication with distance sensor
spi = spidev.SpiDev()
spi.open(0,0)

# open communication with distance sensor
spi = spidev.SpiDev()
spi.open(0,0)

i = 0
r_acc = 0

while True:
  try:

    r_sensor_value = readadc(R_SENSOR_ID)

    if i == NUM_CAMP:

      r_sensor_value = r_acc / NUM_CAMP

      i = 0
      r_acc = 0

      if r_sensor_value >= R_SENSOR_MAX:
        if AUDIO_STARTED == False:
          print("Starting audio...")
          print("read ", r_sensor_value)
          sendUDP('1') # open
          AUDIO_STARTED = True
          run_audio = aplay(AUDIO_NAME, _bg=True)
          R_SENSOR_MAX = R_SENSOR_MAX - 4
    elif r_sensor_value <= R_SENSOR_MIN:
      sendUDP('0') # closed

    else:
      sendUDP('2') # motion

      if AUDIO_STARTED == True:
        AUDIO_STARTED = False
        R_SENSOR_MAX = R_SENSOR_MAX + 4
        print("Stopping audio...")
        print("read ", r_sensor_value)
        run_audio.kill()

  else:
    i += 1
    r_acc += r_sensor_value


  time.sleep(.1)

except KeyboardInterrupt:

  if AUDIO_STARTED == True:
    run_audio.kill()
  sys.exit()
```

Raspi – main_video.py

```
import sys
import time
import re
import argparse
import socket
import pygame
from sh import omxplayer, aplay
import RPi.GPIO as GPIO

# parse arguments
parser = argparse.ArgumentParser()
parser.add_argument("name", help="name of the diary")
parser.add_argument("address", help="ip address of server")
parser.add_argument("port", help="port of server", type=int)
args = parser.parse_args()

VIDEO_NAME = args.name + ".mp4"
AUDIO_NAME = args.name + ".wav"
IMAGE_NAME = args.name + ".jpg"

IMAGE_WIDTH = 1600
IMAGE_HEIGHT = 900

VIDEO_CURR_REXP = re.compile(r'V :\s*([\d.]+).*')
VIDEO_TOTAL_REXP = re.compile(r'Length : *([\d.]+)*')

ADDR = args.address
PORT = args.port

GPIO_SENSORS = [4, 25, 24]

GPIO_PIN = 23

VIDEO_STARTED = False
VIDEO_PAUSED = False

omx_stdin = None
omx_process = None
tot_sec = None
# print splash screen
def print_intro(name):
  print("""        __...--~~~~~~-._  _.-~~~~~~--...__
      //               `V'               \\\
     //                 |                 \\\
    //__...--~~~~~~-._  |  _.-~~~~~~~-...__\\\
   //__....----~~~~~._\ | /_.~~~~-----....__\\\
  ====================\\\|//====================
                      `---`""")
  print("\t[[[[ Diario: " + name.upper() + " ]]]]\n")

# send data to server over UDP protocol
def sendUDP(value):
  sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
  #print('sendUDP ', value)
  sock.sendto(value, (ADDR, PORT))

def setup_buttons():
  GPIO.setmode(GPIO.BCM)
  [GPIO.setup(x, GPIO.IN) for x in GPIO_SENSORS]
  GPIO.setup(GPIO_PIN,GPIO.IN)

def read_buttons(): # get index of button pressed
  button = [x for x in GPIO_SENSORS if not(GPIO.input(x))]
  return 0 if not button else GPIO_SENSORS.index(button[0])+1

# interact with omxplayer STDIN and STDOUT
def interact(line, stdin, process):
  global tot_sec
  global VIDEO_PAUSED
  global omx_stdin
  global omx_process

  omx_stdin = stdin
  omx_process = process

  # get current video time
```

```
    curr = VIDEO_CURR_REXP.search(line)

    if curr and tot_sec:
      pts = curr.group(1)
      sec = int(pts.split(".")[0]) / 1000000

      # stop video to last seconds
      if tot_sec == sec and VIDEO_PAUSED == False:
        VIDEO_PAUSED = True
        stdin.put('p')
        print("---- PAUSE ----")

    else:
      len = VIDEO_TOTAL_REXP.search(line)

      if len:
        tot_pts = len.group(1)
        tot_sec = (int(tot_pts) / 1000) - 13
        # set tot_len to 2 sec before end because of omxplayer buffer

  print_intro(args.name)

  setup_buttons()

  pygame.init()
  pygame.mouse.set_visible(False)

  # init rendering screen
  displaymode = (IMAGE_WIDTH , IMAGE_HEIGHT)
  screen = pygame.display.set_mode(displaymode)

  # load cover image
  cover = pygame.image.load(IMAGE_NAME).convert()

  # set cover position
  position = pygame.Rect((0, 0, IMAGE_WIDTH, IMAGE_HEIGHT))
  screen.blit(cover, position)

  pygame.display.update()

  while True:
    try:

      if VIDEO_STARTED == False:
        if (not(GPIO.input(GPIO_PIN))):
          print("Starting video...")
          sendUDP('1') # open
          VIDEO_STARTED = True
          VIDEO_PAUSED = False
          time.sleep(.5)
          run = omxplayer('-s', VIDEO_NAME, _bg=True, _out=interact, _out_bufsize=250)
          time.sleep(1)
          run_audio = aplay(AUDIO_NAME, _bg=True)
          screen.fill((0,0,0))
          pygame.display.update()

      else:

        # check if chapter button is pressed
        if omx_stdin:
          chapter = read_buttons()

          if chapter > 0:
            if VIDEO_PAUSED == True:
              omx_stdin.put('p')
              print("--- PLAY ---")
              time.sleep(.3)

            run_audio.kill()

            omx_stdin.put("\027[B")
            print('Rewind')
            time.sleep(1)

            if chapter > 1:
              print('sendOMX ' + repr(chapter))
              omx_stdin.put(repr(chapter))
            else:
```

```
          run_audio = aplay(AUDIO_NAME, _bg=True)

        VIDEO_PAUSED = False

    # check if is closed
    if GPIO.input(GPIO_PIN):
      sendUDP('0') # closed
      VIDEO_STARTED = False
      print("Stopping video...")
      omx_stdin = None
      omx_process.kill()
      omx_process.wait()

      run_audio.kill()

      screen.blit(cover, position)
      pygame.display.update()

  pygame.time.delay(100)

except KeyboardInterrupt:

  if VIDEO_STARTED == True:
    omx_process.kill()
    omx_process.wait()
    run_audio.kill()

  pygame.quit()
  GPIO.cleanup()
  #time.sleep(1)
  sys.exit()
```

Arduino Ethernet

```
/*
 * UDPSendReceiveStrings
 * This sketch receives UDP message strings, prints them to the serial port
 * and sends an "acknowledge" string back to the sender
 * Use with Arduino 1.0
 *
 */

#include <SPI.h>         // needed for Arduino versions later than 0018
#include <Ethernet.h>
#include <EthernetUdp.h> // Arduino 1.0 UDP library

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; // MAC address to use
byte ip[] = {192, 168, 2, 201 };    // Arduino's IP address

unsigned int localPort = 8888;      // local port to listen on

const int analogOutPins[] = { 3,5,6 };  // pins 10 and 11 used by ethernet shield

// buffers for receiving and sending data
char packetBuffer[UDP_TX_PACKET_MAX_SIZE]; //buffer to hold incoming packet,
char replyBuffer[] = "acknowledged";       // a string to send back

// A UDP instance to let us send and receive packets over UDP
EthernetUDP Udp;

void setup() {
    // start the Ethernet and UDP:
  Ethernet.begin(mac,ip);
  Udp.begin(localPort);
  pinMode(analogOutPins[0], OUTPUT);
  pinMode(analogOutPins[1], OUTPUT);
  pinMode(analogOutPins[2], OUTPUT);

  Serial.begin(9600);
}

void loop() {
  // if there's data available, read a packet
  int packetSize =  Udp.parsePacket();
  int i = 0;
  if(packetSize)
  {
  //  Serial.print("Received packet of size ");
  //  Serial.println(packetSize);

    // read packet into packetBuffer and get sender's IP addr and port number
    Udp.read(packetBuffer,UDP_TX_PACKET_MAX_SIZE);
   // Serial.println("Contents:");
   // Serial.println(packetBuffer);

    int pin = packetBuffer[0]-48;
    int value = packetBuffer[2]-48;
    Serial.println(pin);
    Serial.println(value);

    if (pin < 2) {
      if (value == 0) {
        analogWrite(analogOutPins[pin], 0);
      }else if (value == 2) {
        for (i = 0; i < 256; i++) {
          analogWrite(analogOutPins[pin], i);
          delay(10);
        }
      }
    }
    //  analogWrite(analogOutPins[packetBuffer], value);

    // send a string back to the sender
    //Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
    //Udp.write(replyBuffer);
    //Udp.endPacket();
  }
  delay(10);
}
```

RaspiPI – config.conf

```
DIARIO=contessa_emilia
```