R A  S P Y

Alexandra Rodríguez Ramos

Proyecto 2º ASIR (2015)

ÍNDICE

1. Introducción

a. Objetivo y justificación

En los últimos años, durante el periodo de crisis, el robo en domicilios ha aumentado considerablemente llevando consigo más inseguridad para el ciudadano. Igualmente, ha aumentado la ocupación de viviendas vacías por parte de terceros.

También, durante la crisis, ha aumentado el paro, lo que ha provocado un aumento de hogares con más de una unidad familiar desempleadas llegando a haber hogares sin ninguna miembro familiar, en edad laboral, trabajando. Según datos de http://www.expansion.com/economia/2015/04/12/552a7fa2268e3e4f578b4578.html existen 1.485.700 personas que viven en hogares sin ingresos.

Todas estas situaciones conllevan a un aumento de desconfianza e inseguridad ciudadana por lo que muchos ha tomado medidas preventivas para evitar este tipo de situaciones. La mayoría instala puertas blindadas, lo que conlleva a dificultar que se produzcan estos hechos delictivos. Son los que menos, los que optan por instalar sistemas de vigilancia ya sea por elevado coste o desconocimiento del servicio.

Lo que se pretende conseguir con este proyecto es que la gente disponga de un servicio de vigilancia de bajo coste, con el que se puedan sentir seguros a la par de protegidos en sus hogares y demás entornos.

b. Análisis de lo existente

Existe en la actualidad diversas empresas que se dedican a la instalación y mantenimiento de sistemas de seguridad en el hogar, donde dependiendo de ciertos factores como la situación geográfica, tipo de vivienda…. el presupuesto varía.

También nos podemos encontrar en diversos comercios sistemas de vigilancia, los cuales podemos instalar nosotros. Estos comercios, como Leroy Merlin, PCComponentes… venden el pack a partir de 160€, dependiendo de los componentes que lo formen, el importe puede aumentar considerablemente hasta el doble.

En definitiva, existe diversidad de productos que ofrecen vigilancia en el hogar a distintos precios, de los cuales, algunos llevan mantenimiento y contacto 24 horas con el cliente y aviso a las fuerzas de seguridad en caso de que ocurra algún incidente.

c. Propuesta detallada

Raspy es un sistema de vigilancia para el hogar. En él se incluyen: una webcam para tener acceso a la zona vigilada en tiempo real; un sensor: lo que hace es detectar movimiento y enviar un correo en caso de que detectase algún movimiento dentro del rango de acción; un teclado: que básicamente sirve para activar y desactivar la alarma (el sensor).
De este modo queremos tener a nuestra disposición un sistema de vigilancia a bajo coste y que nos ofrezca una seguridad.

2. Planificación temporal y evaluación de costes

Para realizar la planificación temporal, se ha tenido en cuenta el tiempo necesario desde la planificación del proyecto (consultando los productos que se van a utilizar), hasta la disposición de ellos en el domicilio y su puesta a punto.

La planificación la podemos dividir en los siguientes puntos:

- **Análisis**: Este punto está enfocado en buscar cual va a ser la finalidad del proyecto y sus posibles usos.

- **Diseño**: Una vez que ya sabemos cuál es la finalidad del proyecto, diseñamos un prototipo de lo que va a ser el proyecto, donde se incluyen todos los componentes necesarios para el funcionamiento del dispositivo y los programas que va a ser necesarios crear para el correcto funcionamiento. En este apartado también se incluye el diseño de la maqueta donde irán incluido todos los componentes.

- **Pruebas**: En este punto se incluyen las pruebas que se van realizando con cada componente, tanto con el programa necesario para su funcionamiento como la disposición del componente en la placa.

- **Implantación**: Finalmente, después de realizar todas las pruebas oportunas y comprobar el correcto funcionamiento, se dispone a unificar todos los programas y componentes en la placa y en la maqueta.

Para la adquisición de los productos, se ha contado con varios proveedores, buscando siempre ofertas, buenos precios y la rápida disposición de los productos necesarios para dicho proyecto, buscando minimizar costes y por lo tanto abaratar el servicio.

COMPONENTE	PROVEEDOR	CANTIDAD	PRECIO FINAL
Raspberry pi Modelo B	pccomponentes	1	29.95
Cámara web	Media markt	1	16.95
Sensor pir	Amazon italia	1	4.20
Pack jumpers	inven	75	3.50
Teclado membrana 4x3	inven	1	2.95
Placa protoboard	pccomponentes	1	3.25
Materiales maqueta	Papeleria de barrio		9.80

En algunos componentes se ha hecho necesario adquirir más de una unidad por problemas técnicos, al conectar mal los pines y fundirse el componente. Habiendo adquirido en un primer momento una única unidad y falta de disponibilidad para segundo pedido en ese proveedor.

El tiempo final aproximado utilizado para la realización de dicho proyecto es de aproximadamente **250 horas** (incluyendo las horas necesarias para buscar en internet los componentes), más aparte el tiempo necesario de espera para la recepción de componentes que al ser en distinta fecha e ir disponiendo de ellos poco a poco, se ha podido ir trabajando durante el proceso de desarrollo. También se incluye en este punto el tiempo para realizar la maqueta.

El precio final asciende a **70.60€** Teniendo en cuenta únicamente los componentes utilizados y la maqueta. Ni equipos informáticos ni router ni cables de internet se han cuantificado ya que ya se disponía de ellos anteriormente.

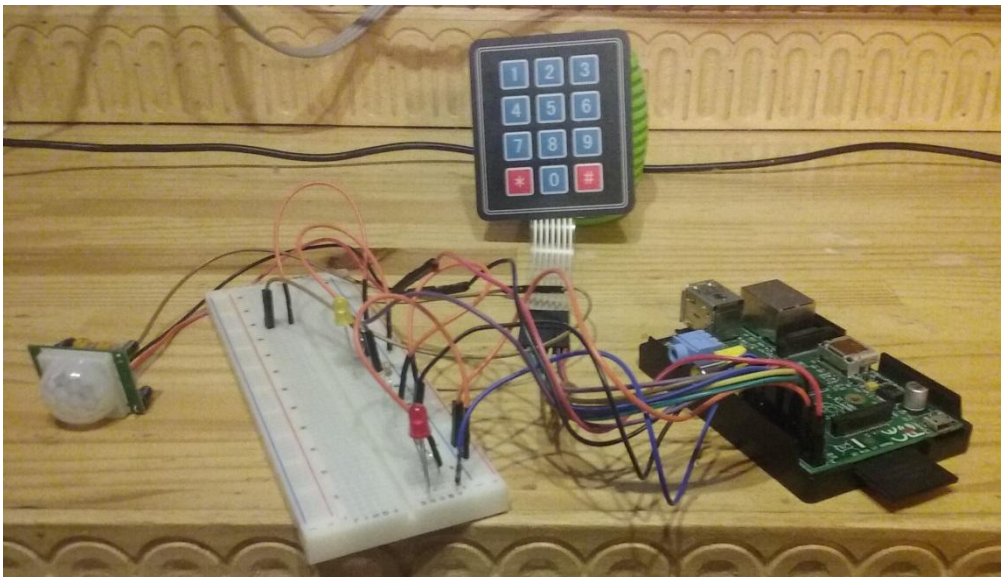
3. Análisis de requisitos

Los requisitos que debe cumplir el equipo son los suficientes para su implantación. Existe actualmente en el mercado las ampliaciones necesarias para el correcto funcionamiento del sistema. Para el correcto funcionamiento de las ampliaciones y del código, es necesario que esté conectado a una red de internet y que se disponga del navegador "FIREFOX" para poder visualizar el video en directo de la webcam.

Los requisitos que se deben dar en el entorno, deben ser los favorables para el correcto funcionamiento del equipo evitando temperaturas extremas, meteorología desfavorable…

4. Diseño

En este punto se especifican las conexiones que se han realizado de los diferentes accesorios a los pines de la Raspberry. La conexión realizada a la protoboard desde el pin “Ground”, conecta varios dispositivos: leds (amarillo y rojo) y el sensor pir.



En esta imagen se puede observar los diferentes componentes que forman el proyecto: la raspberry, teclado, sensor pir , la protoboard y los led conectados a ésta última que, al no disponer de pantalla, indican si la alarma ha sido conectada o desconectada.

5. Codificación

a. Entorno de programación

Se ha programado en Python dentro del editor de textos del propio sistema de la Raspberry.

b. Lenguajes y herramientas

En este caso, nos hemos servido del lenguaje de programación Python.
Se ha programado en el propio sistema y se han incorporado al inicio del equipo para que arranque según se inicie el sistema, de este modo, la persona únicamente se debe preocupar de tener el sistema conectado a una red eléctrica.

c. Aspectos relevantes de la implantación.

Se dispone de varios códigos según la función a realizar.

- SENSOR PIR
- WEB CAM
- TECLADO

Al no disponer de pantalla conectada directamente al sistema, para poder verificar que la alarma se ha conectado/desactivado correctamente, disponemos de unos led con los que se verifica el correcto funcionamiento de la alarma.

El acceso al programa está restringido a las personas que tengan acceso al sistema y a la red.

Como el programa que se ha creado se inicia directamente con el arranque del equipo, se han incluido las siguientes líneas para que se dé tal hecho:

```
#!/usr/bin/python
### BEGIN INIT INFO
# Provides: archivo proyecto
# Required-Start: $syslog
# Required-Stop: $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: archivo proyecto
# Description:
#
### END INIT INFO
```

6. Pruebas de ejecución

Una vez que se ejecutan los scripts utilizados junto con el inicio del sistema cuando se arranca el equipo, la aplicación se ejecuta automáticamente junto con el resto del sistema. Por lo que las pruebas de ejecución se realizan según se encienda la placa.

Al no tener una pantalla donde podamos comprobar que se enciende y apaga la alarma, tenemos unos led que nos lo verifican. Para proteger los leds y la placa, se ha dispuesto de unas resistencias para protegerlos de subidas de tensión.

7. Manuales de usuario

a. Objetivo

El objetivo del manual es que el cliente comprenda y use correctamente el producto y poder solucionar posibles errores o problemas que puedan surgir durante su uso.

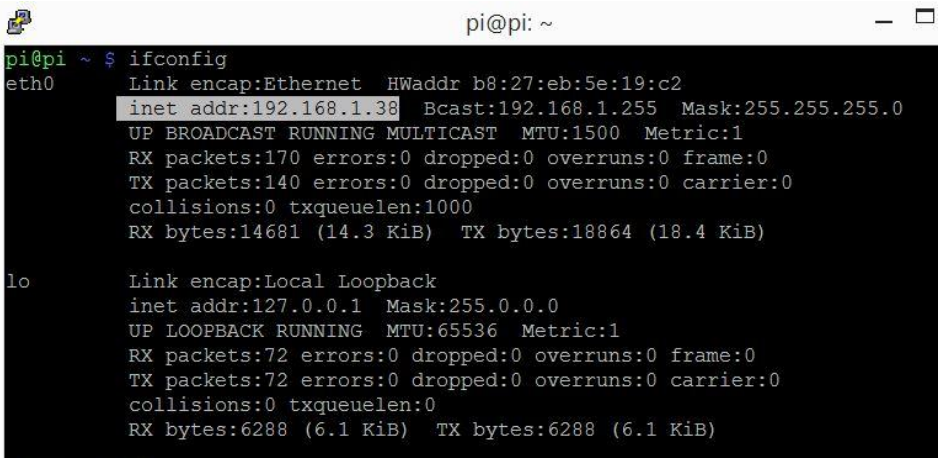
b. Requisitos e instalación

Para el correcto funcionamiento del equipo, es necesario que se cumplan una serie de requisitos:

- Tener el equipo conectado a una conexión eléctrica.
- No forzar el apagado del sistema.
- No desconectar ningún componente si el equipo sigue conectado a la red eléctrica.
- No exponerlo a situaciones extremas tanto climatológicas como de entorno para evitar que se produzcan fallos irreversibles en el sistema.

La instalación es básica, únicamente hay que conectar todos los componentes al equipo, conectar el sistema a una corriente eléctrica y arrancar el sistema.

La primera conexión, es necesario hacerla con una pantalla, una vez conectado la placa a internet. Desde la pantalla y una vez iniciada la raspberry, abrimos el terminal y ponemos "IFCONFIG" lo que nos da la ip del equipo para poder visualizar luego el video de la webcam". Una vez que se haya realizado los procesos de inicio del equipo, ya empieza a funcionar.



Dirección ip de la Raspberry

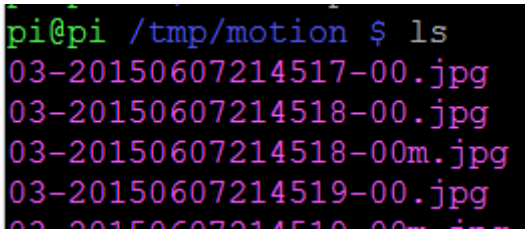
c. Descripción del funcionamiento del sistema

El arranque de los programas creados se hace junto con el arranque del equipo, por lo que en cuanto se inicia el equipo, los programas se arrancan y ya no habría modificar nada.

El acceso de la webcam se podría hacer directamente a través del navegador Firefox. Simplemente habría que escribir en el navegador la dirección IP del equipo:puerto (al que esté conectado el dispositivo) y ya se dispondría de acceso a la imagen en directo que capta la cámara. Para saber la ip de la raspberry, necesitamos conectarla a una pantalla a través de cable HDMI y en el terminal escribir “ifconfig”, lo que nos dará la dirección ip de la raspberry.

El resto de componentes se detecta automáticamente una vez iniciado el equipo y automáticamente el sensor, en cuanto detecte movimiento, avisaría a la persona con un correo electrónico.

Se ha configurado la webcam para que guarde las imagenes de antes y despues de que detecte el movimiento, por lo que vamos a disponer de ellas cuando las necesitemos. La carpeta de acceso a ellas está en /tmp/motion.



d. Mensajes de error

En un principio no debería tener mensajes de error. Pero se podrían dar los siguientes casos:

- El navegador no detecta la imagen de la webcam: En este caso hay que asegurarse que la webcam esté bien conectada, asegurarse de que el navegador utilizado es el Firefox y que dispone de conexión a internet.

- No dispone de conexión a internet: Puede ser que la conexión no se haya realizado. Por lo que hay que asegurarse que si está conectado por cable o wifi, que el router esté conectado y funcionando. En caso de estar conectado por wifi, asegurarse que equipo se encuentre dentro del rango de detección del router o en su defecto, amplificador del wi-fi que se haya podido instalar.

  En el caso de estar conectados a través de modem usb, hay que comprobar que está conectado al puerto usb correctamente y en caso de que no disponga de internet, comprobar que se disponga de datos de internet en la tarjeta .

- La Raspberry no se inicia: cuando se vaya a conectar la raspberry a la red eléctrica y se vea que no funciona, es necesario desconectarla de la red eléctrica, quitar y volver a poner la tarjeta sd (puede que al estar mal conectada, no la pueda leer y no se inicie) y volver a conectarla a un enchufe o dispositivo eléctrico.

8. Conclusiones

e. Conclusiones sobre el trabajo realizado

Para poder realizar el proyecto he procurado tener algún componente disponible para ir trabajando y así no pasar tiempo sin poder hacer nada por falta de disponibilidad de componente.

Ha sido un trabajo complejo ya que he utilizado componentes y una metodología que no había utilizado anteriormente, pero buscando en manuales, internet y demás, el trabajo se ha podido facilitar. He podido probar diferentes componentes y códigos y ver el proceso que se realizaban

con ellos, ajustándolos, en la medida de lo posible, al proyecto que he realizado.

f. Conclusiones sobre el sistema desarrollado

Para desarrollar el proyecto se ha utilizado código Python dentro del sistema operativo de Linux, Raspbian.

El hecho de utilizar código Python que no había utilizado nunca, ha hecho que se haya complicado un poco más el proyecto y haya sido necesario utilizar más horas para buscar códigos que funcionasen, poder entender cual era el funcionamiento que realizaban y en la medida de lo posible, modifcarlos para poder utilizarlos en el proyecto.
También el hecho de utilizar componentes eléctricos, que no había utilizado, ha supuesto un desafío, ya que nunca lo había utilizado junto con una raspberry.

En conjunto, el desarrollo del programa con Python y el uso de componentes eléctricos para la raspberry, han supuesto una novedad, y aún después de algún "susto", he conseguido entender el funcionamiento y la metodología que conlleva su uso.

g. Conclusiones personales

Es un proyecto que tenía ganas de hacer desde hace tiempo. Me ha llevado muchas horas el tema de buscar componentes que quería utilizar, ya que quería encontrar componentes que económicamente fueran asequibles (había sitios donde el precio se llegaba a triplicar) y que fuera fácil y rápido disponer de ellos. También el buscar cómo crear el código para los distintos componentes y que hicieran lo que buscaba me ha llevado muchas horas. En un momento, ha sido imposible conectar algún componente (MODULO NFC) que quería utilizar ya que tuve un problema con él (no sé si error de código, de ejecución, problema físico del módulo...) por lo que tuve que descartarlo y buscar algo que se asemejara para realizar la misma función.

Llegó un momento en el que se quemó un sensor y no disponía de más, tuve problemas para conseguirlo donde lo adquirí en un primer momento y tuve

que pedirlo por internet fuera de España (estábamos en plena semana santa) para poder disponer de él lo antes posible, este hecho trastocó la planificación.

Aún con todos los problemas que me he ido encontrando por el camino, me ha resultado grato hacer este proyecto y ver cómo iba avanzando y consiguiendo cosas empezando desde 0 en algunos campos.

A través de prueba/error, he ido aprendiendo, lo que ha llevado consigo que fuera aumentando las ganas de mejorar y hacerlo con más ganas el proyecto.

Ya que no quería que fuera muy alto el presupuesto y por falta de tiempo a la hora de buscar el código necesario, no he incluido varios componentes que le podrían haber dado otro atractivo al proyecto. Aunque prácticamente he utilizado todos los pines y apenas podría haber desarrollado el proyecto con algún componente más.

Estoy contenta con el trabajo desarrollado aunque se podría haber mejorado, pero no descarto implantar el proyecto, realizando algunos cambios, en mi vivienda.

En definitiva, estoy contenta por haber podido hacer el proyecto que quería hacer desde hace tiempo aunque se podría mejorar.

h. Posibles ampliaciones y mejoras

El proyecto desarrollado cuenta con una gran variedad de mejoras que se podrían añadir al proyecto disponiendo de un tiempo adecuado, un mayor presupuesto y dependiendo del entorno donde se vaya a instalar. Pasamos a enumerar las posibles mejoras dependiendo del entorno de implementación:

- **LECTOR DE HUELLA DIGITAL**: En el caso de un entorno donde requiera mayor seguridad para que únicamente las huellas que se encuentren registradas, puedan interactuar con la máquina.

- **PLACA SOLAR CON BATERIA**: En entornos donde no es posible conectar el equipo a la red eléctrica o que se necesite que sea independiente. De este modo, el equipo podría funcionar las 24 horas del día.

- **WEB CAM DE INFRARROJOS**: En entornos donde apenas se disponga de iluminación y se vayan a hacer grabaciones o recibir imagen en

zonas con visibilidad reducida y así no perder detalle de lo que ocurra.

- **MODEM USB**: Para el caso en el que no se disponga de una conexión a internet. En el modem es posible insertar una tarjeta SIM de teléfono y contratando datos es posible tener conexión a internet.

- **ADAPTADOR WIFI-USB**: para el caso en el que no se disponga de una conexión a internet por cable o no sea factible conectar el cable.

- **MODULO NFC:** disponible para desactivar/activar el dispositivo. Se utiliza con tarjetas disponibles por internet e incluso con un Smartphone que disponga de tal modulo.

- **PANTALLA:** Al disponer de una pantalla, se puede interactuar insitu con los programas disponibles y comprobar las modificaciones que se realizan.

9. Bibliografía

DÓMOTICA

- Beginning NFC EDITORIAL: O'reilly
- Building a home security system with beaglebone EDITORIAL: Packt publishing
- RFID Handbook AUTOR: Klaus Finkenzeller
- Programming your home AUTOR: Mike Riley
- Raspberry cookbook AUTOR: O'reilly

PYTHON

- http://www.cursosdeprogramacionadistancia.com/static/pdf/material-sin-personalizar-python.pdf

MODULO NFC

- http://fuenteabierta.teubi.co/2013/07/utilizando-el-lector-nfc-rc522-en-la.html
- http://panamahitek.com/mifare-rc522-arduino/
- http://geraintw.blogspot.com.es/2014/01/rfid-and-raspberry-pi.html
- https://gist.github.com/mattgorecki/6085344
- http://translate.google.es/translate?hl=es&sl=de&u=http://www.elektronx.de/tutorials/rfid-tags-auslesen/&prev=search

- http://translate.google.es/translate?hl=es&sl=en&u=http://www.pridopia.co.uk/pi-rfid-reader.html&prev=search
- https://learn.adafruit.com/babel-fish/programming
- http://translate.google.es/translate?hl=es&sl=en&u=http://bsd.ee/~hadara/blog/%3Fp%3D1017&prev=search

SENSOR PIR

- http://www.internetdelascosas.cl/2013/05/13/sensor-de-presencia-en-raspberry-pi/
- http://www.electronicaembajadores.com/Productos/Detalle/37/SSPIR01/sensor-pir-de-movimiento
- https://www.raspberrypi.org/learning/parent-detector/worksheet/
- http://www.raspberrypi-spy.co.uk/2013/02/cheap-pir-sensors-and-the-raspberry-pi-part-2/
- http://www.inven.es/es/sensores/78-sensor-de-movimiento-pir-hc-sr501.html
- http://www.repparts3d.com/robotica-diy/211-sensor-infrarrojos-hc-sr501.html

TECLADO

- http://rpiplus.blogspot.com.es/2013/06/teclado-hexadecimal-4x4.html
- http://www.embarcados.com.br/raspsberrypi-teclado-4x4-com-python/

LED

- https://projects.drogon.net/raspberry-pi/gpio-examples/tux-crossing/gpio-examples-1-a-single-led/
- https://geekytheory.com/tutorial-raspberry-pi-gpio-parte-2-control-de-leds-con-python/
- http://fpaez.com/como-controlar-un-led-con-raspberry-pi-y-pwm/

10. Apéndices

i. Codígo webcam

```
pi@pi ~ $ sudo apt-get install motion
```

```
  GNU nano 2.2.6                Fichero: /etc/default/motion

# set to 'yes' to enable the motion daemon
start_motion_daemon=yes
```

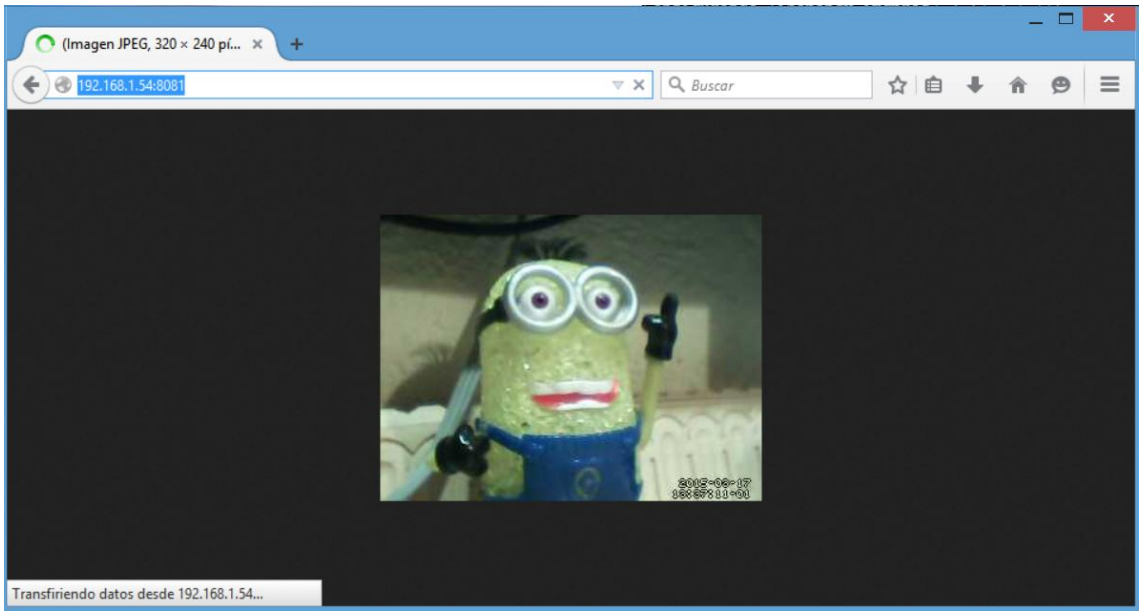
/etc/motion/motion.conf

```
# Start in daemon (background) mode and release terminal (default: off)
daemon on
```

```
############################################################
# Live Webcam Server
############################################################

# The mini-http server listens to this port for requests (default: 0 = disabled)
webcam_port 8081

# Quality of the jpeg (in percent) images produced (default: 50)
webcam_quality 50

# Output frames at 1 fps when no motion is detected and increase to the
# rate given by webcam_maxrate when motion is detected (default: off)
webcam_motion on

# Maximum framerate for webcam streams (default: 1)
webcam_maxrate 1

# Restrict webcam connections to localhost only (default: on)
webcam_localhost off
```

```
############################################################
# HTTP Based Control
############################################################

# TCP/IP port for the http server to listen on (default: 0 = disabled)
control_port 8080

# Restrict control connections to localhost only (default: on)
control_localhost off

# Output for http server, select off to choose raw text plain (default: on)
control_html_output on

# Authentication for the http based control. Syntax username:password
# Default: not defined (Disabled)
; control_authentication username:password
```

```
# Specifies the number of pre-captured (buffered) pictures from before motion
# was detected that will be output at motion detection.
# Recommended range: 0 to 5 (default: 0)
# Do not use large values! Large values will cause Motion to skip video frames and
# cause unsmooth mpegs. To smooth mpegs use larger values of post_capture instead.
pre_capture 1

# Number of frames to capture after motion is no longer detected (default: 0)
post_capture 1
```



Se ha configurado para que se guarden imagenes de antes y después de que detecte movimiento. La carpeta donde se encuentran dichas imagenes es en : /tmp/motion.
El formato de las imagenes es:

```
pi@pi /tmp/motion $ ls
03-20150607214517-00.jpg
03-20150607214518-00.jpg
03-20150607214518-00m.jpg
03-20150607214519-00.jpg
```

03-AÑO-MES-DIA-HORA-MINUTOS-SEGUNDOS-00.JPG de este modo, sabemos que ha pasado por delante de la web cam cuando se ha detectado movimiento con ella.

j. <u>Código python unificado de todos los componentes.</u>

```
import RPi.GPIO as GPIO
import time
import smtplib
import email.utils


GPIO.setmode(GPIO.BCM)
#VALOR VARIABLES
rows = [18, 17, 27, 23]
cols = [22, 24, 25]
keys = [
   ['1', '2', '3'],
   ['4', '5', '6'],
   ['7', '8', '9'],
   ['*', '0', '#']]
amarillo=8
rojo=11
sensor=4
#TIPO DE DATO: ENTRADA/SALIDA
GPIO.setup(amarillo, GPIO.OUT)
GPIO.setup(sensor, GPIO.IN)
GPIO.setup(rojo, GPIO.OUT)

#CODIGO TECLADO
for row_pin in rows:
   GPIO.setup(row_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

for col_pin in cols:
   GPIO.setup(col_pin, GPIO.OUT)

def get_key():
   key = 0
   for col_num, col_pin in enumerate(cols):
      GPIO.output(col_pin, 1)
      for row_num, row_pin in enumerate(rows):
         if GPIO.input(row_pin):
            key = keys[row_num][col_num]
      GPIO.output(col_pin, 0)
   return key
```

```
while True:
   key = get_key()
   if key :
      print(key)
   time.sleep(0.3)

   encender = get_key()
   apagar = get_key()



#CODIGO TECLADO. ACTIVAR ALARMA

   if encender == "8":
      print("alarma encendida")
      time.sleep(1)
      GPIO.output(amarillo,GPIO.HIGH)
      time.sleep(1.5)
      GPIO.output(amarillo,GPIO.LOW)
      time.sleep(1)

         while True:
#codigo sensor de movimiento
                  input_state = GPIO.input(sensor)
                  if input_state == True:
                           print("movimiento detectado")
                           time.sleep(2)

#codigo correo electronico
                           GMAIL_USER = 'AQUI VA LA CUENTA DE CORREO'
                           GMAIL_PASS = 'AQUI VA LA CONTRASEÑA'
                                    SMTP_SERVER = 'smtp.gmail.com'
                           SMTP_PORT = 587

                           def send_email(recipient, subject, text) :
                                    smtpserver   =   smtplib.SMTP(SMTP_SERVER,
SMTP_PORT)
                  smtpserver.ehlo()
                                    smtpserver.starttls()
                                    smtpserver.ehlo
                                    smtpserver.login(GMAIL_USER, GMAIL_PASS)
                  header = 'to:' + recipient + '\n' + 'from:' + GMAIL_USER
                  header = header + '\n' + 'subject ' + subject + '\n'
                                       msg = header + '\n' + text + '\n\n'
```

```
                                        smtpserver.sendmail(GMAIL_USER, recipient,
msg)
                                                smtpserver.quit()
                                                send_email('CORREO ELECTRONICO', '
', 'se ha detectado movimiento')
                                print("correo enviado")

#CODIGO TECLADO. DESACTIVA ALARMA. BREAK ROMPE EL CICLO DE
DETECTAR MOVIMIENTO
#CUANDO SE ACTIVA LA TECLA "9"
                        if get_key() == "9":
                                break
                print("alarma apagada")
#ENCIENDE Y APAGA UNA SOLA VEZ UN LED
                time.sleep(1)
                GPIO.output(rojo,GPIO.HIGH)
                time.sleep(1.5)
                GPIO.output(rojo,GPIO.LOW)
                time.sleep(1)


GPIO.cleanup()
```

k. CÓMO EJECUTAR EL SCRIPT CON EL ARRANQUE DEL EQUIPO

- Damos permisos al archivo

```
pi@pi ~ $ sudo chmod 777 proyectoS.py
```

- Copiamos el archivo a la carpeta /etc/init.d/

```
pi@pi ~ $ sudo cp proyectoS.py /etc/init.d/
```

- Hacemos que se ejecute al inicio del sistema

```
pi@pi ~ $ sudo update-rc.d proyectoS.py defaults
```

- Reiniciamos el sistema y lo probamos

I. Código Python con todo incluido.

```
#!/usr/bin/python
### BEGIN INIT INFO
# Provides: archivo proyecto
# Required-Start: $syslog
# Required-Stop: $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: archivo proyecto
# Description:
#
### END INIT INFO


import RPi.GPIO as GPIO
import time
import smtplib
import email.utils
#import os

GPIO.setmode(GPIO.BCM)

rows = [18, 17, 27, 23]
cols = [22, 24, 25]
keys = [
    ['1', '2', '3'],
    ['4', '5', '6'],
    ['7', '8', '9'],
    ['*', '0', '#']]
amarillo=8
rojo=11
sensor=4
GPIO.setup(amarillo, GPIO.OUT)
GPIO.setup(sensor, GPIO.IN)
GPIO.setup(rojo, GPIO.OUT)

for row_pin in rows:
    GPIO.setup(row_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

for col_pin in cols:
    GPIO.setup(col_pin, GPIO.OUT)

def get_key():
    key = 0
    for col_num, col_pin in enumerate(cols):
        GPIO.output(col_pin, 1)
```

```
            if GPIO.input(row_pin):
                key = keys[row_num][col_num]
        GPIO.output(col_pin, 0)
    return key

while True:
    key = get_key()
    if key :
        print(key)
    time.sleep(0.3)

    encender = get_key()
    apagar = get_key()




    if encender == "8":
 #       print("alarma encendida")
        time.sleep(1)
        GPIO.output(amarillo,GPIO.HIGH)
```

```
      time.sleep(1.5)
      GPIO.output(amarillo,GPIO.LOW)
      time.sleep(1)

      while True:
              input_state = GPIO.input(sensor)
              if input_state == True:
                      print("movimiento detectado")
                      time.sleep(2)

                      GMAIL_USER = 'sandra.correo.proyecto@gmail.com'
                      GMAIL_PASS = 'alexandra0488'
                      SMTP_SERVER = 'smtp.gmail.com'
                      SMTP_PORT = 587

                      def send_email(recipient, subject, text) :
                              smtpserver = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
                              smtpserver.ehlo()
                              smtpserver.starttls()
                              smtpserver.ehlo
                              smtpserver.login(GMAIL_USER, GMAIL_PASS)
                              header = 'to:' + recipient + '\n' + 'from:' + GMAIL_USER
                              header = header + '\n' + 'subject ' + subject + '\n'
```

```
                              msg = header + '\n' + text + '\n\n'
                              smtpserver.sendmail(GMAIL_USER, recipient, msg)
                              smtpserver.quit()
                      send_email('sandra.correo.proyecto@gmail.com', ' ', 'se ha detectado movimiento')
#                     print("correo enviado")
              if get_key() == "9":
                      break
#       print("alarma apagada")
        time.sleep(1)
        GPIO.output(rojo,GPIO.HIGH)
        time.sleep(1.5)
        GPIO.output(rojo,GPIO.LOW)
        time.sleep(1)

GPIO.cleanup()
```