SISTEMA DE MONITOREO REMOTO BASADO EN IoTs

Jaime Andrés Fernández González
Hugo Mario Zuleta Ortega

Asesor: Yezid Donoso Meisel

PROYECTO DE GRADO
PREGRADO INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
UNIVERSIDAD DE LOS ANDES
Semestre Académico 2015-2

Contenido

Introducción 3
Descripción del problema y soluciones disponibles 3
Diseño e Implementación de la solución 3
Trabajo futuro 8
Bibliografía 9
Apéndice A – Código Fuente sin formato 10

Introducción

En este documento se aduce un proyecto de grado en el cual se aborda el problema de la seguridad doméstica por medio de una solución ad-hoc de bajo costo que permite el monitoreo de un área cualquiera dentro de un domicilio.

Dicha solución permite la transmisión de video en vivo y captura de video que se hace llegar a un usuario en el momento en el que se detecta movimiento en el área. La implementación se llevó a cabo usando una tarjeta Raspberry Pi, una cámara web convencional, un sensor pasivo infrarrojo, y una conexión a internet vía Ethernet. En particular, el sistema permite monitorear el área en tiempo real dentro de una red local (LAN) y si detecta movimiento: graba un video del posible intruso, sube el video al servicio web de almacenamiento de video Gfycat y finalmente envía al usuario un mensaje directo de Twitter con el enlace al video en formato "webm".

El documento se divide en tres principales secciones: una contiene detalles del tópico generador del proyecto, otra contiene detalles de su implementación, y finalmente se dedica una sección a unas posibles maneras en las cuales el proyecto se puede extender, en miras al trabajo futuro.

Descripción del problema y soluciones disponibles

El problema es la inseguridad en locales comerciales y residencias, en las cuales, según cifras de la Dijin del 2013, hubo más de 43.000 robos perpetuados en horas laborales (entre las 12m y 6pm) [1] [2]. Adicionalmente el DANE publicó que en 2014 el 38% de los hogares colombianos tenían acceso a internet, con estimaciones de alcanzar un 42% de mantenerse la tasa de crecimiento [3].

Si bien, en este marco se presentan soluciones de monitoreo doméstico en el mercado, como lo son los productos "Canary" y "Piper" que ofrecen software robusto y hardware de alta calidad, sus costos son mayores que el precio que estaría dispuesta a pagar una familia colombiana promedio ($199USD por Canary y $279USD por Piper al momento de la presentación de este documento) y su capacidad de expansión heterogénea, i.e. adicionar sensores distintos a los ofrecidos o poder monitorear varias habitaciones al tiempo, es muy baja [4] [5].

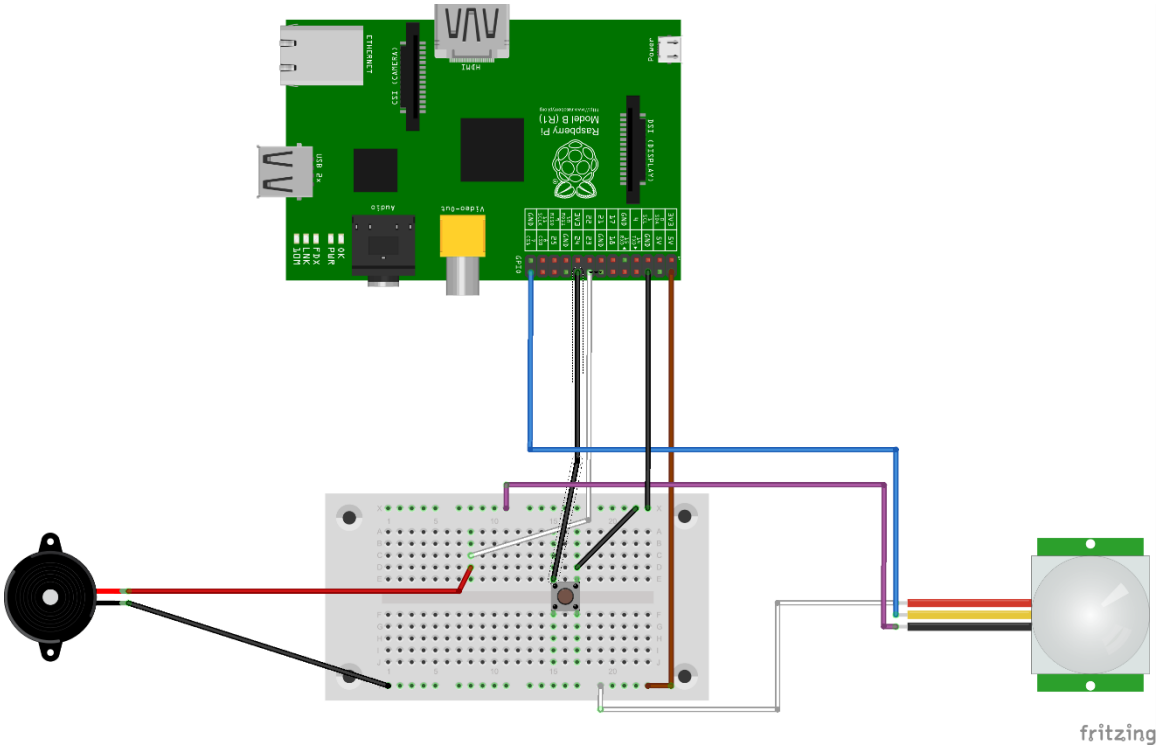
Surge entonces la necesidad de un sistema de monitoreo de bajo costo y que mitigue los riesgos de dejar el hogar abandonado para realizar las labores diarias.

Diseño e Implementación de la solución

La solución está construida sobre una Raspberry Pi (Model B) que tiene instalada imagen de Raspbian disponible al momento (Linux raspberrypi 3.18.11+). Los demás elementos que componen la solución son:

		
Cámara web Logitech C310	Zumbador electrónico de sonido continuo	Módulo sensor de movimiento PIR (pyroelectric infrared)

La cámara web se conecta vía USB, mientras que los otros componentes de conectan siguiendo el siguiente esquema que hace uso de una protoboard y un pulsador que hará las veces de interruptor:



Posteriormente se crea un archivo de extensión .py y se pone el código fuente, el cual se explica por sí mismo con excepción de algunas líneas (se añade una copia sin formato en el Apéndice A para facilitar su reproducción):

```
1.  #!/usr/bin/env python
```

```
2.  import RPi.GPIO as GPIO
3.  import datetime
4.  import time
5.  import os, errno
6.  import tweepy
7.  import threading
8.  import subprocess
9.  import base64
10. import json
11. import requests
12. from base64 import b64encode
13. from client import GfycatClient
14.
15.
16. GPIO.cleanup()
17.
18. led = 4              #on-
    board LED pin number. Edit depending on your Raspberry Pi configuration
19. default_dist = 0        #normal distance to opposite wall
20. dist0 = 7           #GPIO pin number for distance sensor (analog ultrasound sens
    or)
21. buzzer = 23         #GPIO pin number for the buzzer
22. button = 24         #GPIO pin number for the button
23. on = True           #boolean that indicates whether the system is armed
24. photos = "./webcam/"
25. currTime = 0            #represents the current Datetime object
26.
27. #tweepy local variables
28. #twitter handle is RPipg (RaspberryPiproyectodegrado), pw:raspberrypi
29. #Access Level: Read, write, and direct messages
30. #keys and access tokens generated on September 27, 2015
31. #apps.twitter.com
32. apikey = "cuPArm9C54e7S6MCMHjE8xLBu"
33. apisecret = "oPPDyRFACPy0mdSL3DCrfhkryY7nc5KGZDY4Yes69pFazzQZVo"
34. accesstoken = "3799681336-N73XQxwTBskGelccSg2FfKO8A6Cw9APvjhBGbgu"
35. accesstokensecret = "sjXPUziK34vUPDNLrcxuftAzGX9MCi8iYUJr3bgujfojM"
36. auth = tweepy.OAuthHandler(apikey, apisecret)
37. auth.secure = True
38. auth.set_access_token(accesstoken, accesstokensecret)
39. api = tweepy.API(auth)
40.
41. print "Started\n"
42.
43. #setting up the GPIO ports
44. GPIO.setmode(GPIO.BCM)
45. GPIO.setwarnings(False)
46.
47. #setup for the on-board LED
48. GPIO.setup(led, GPIO.OUT)
49.
50. #this signals the program will start
51. #turns on the on-board LED three times, and leaves it on the last time
52. GPIO.output(led, 1)
53. time.sleep(1)
54. GPIO.output(led, 0)
55. time.sleep(1)
56. GPIO.output(led, 1)
57. time.sleep(1)
58. GPIO.output(led, 0)
59.
60. #setup for the distance sensors, readings in cm
```

```
GPIO.setup(dist0, GPIO.IN)

#setup for the button
GPIO.setup(button, GPIO.IN)#, pull_up_down = GPIO.PUD_DOWN)

#setup for the buzzer
GPIO.setup(buzzer, GPIO.OUT)

#starts the entire security monitoring script
def main():
    #This script does the following:
    #   -Start the live feed though LAN
    #   -If dist is reduced significantly,
    #       the alarm will be sent to the user, with a video of the scene, and
    #       the board will make noise.
    #   -
Take a video when the alarm is triggered. It will be saved locally and
    #       sent to Gfycat.
    #   -
Tweet the alert. Send a PM to the user with the link to the Gfycat video
    try:
        global on
        check_button_thread()
        #start the local webcam server
        subprocess.Popen("sudo service motion start", shell=True)
        print "Started webcam server."
        #connect to server and load settings from User

        #create the folder where the pictures will be saved
        mkdir_p(photos)
        current_state = 0
        previous_state = 0

        print "waiting for sensor to settle"
        while GPIO.input(dist0) == 1:
            current_state = 0

        #start checking on the sensors indefinitely
        while True:
            if on:
                print "Sensor reading: "
                current_state = GPIO.input(dist0)
                if current_state == 1 and previous_state == 0: #something approa
Ched the sensor
                    print "!! Possible intruder. Taking action...\n"
                    make_sound_thread()
                    #stop the webcam server daemon
                    subprocess.check_call("sudo service motion stop", shell=True
)
                    print "Stopped webcam server."
                    link = take_video()
                    #restarts the LAN webcam server
                    subprocess.check_call("sudo service motion start", shell=Tru
e)
                    print "Restarted webcam server."
                    alert_user(link)
                    time.sleep(1)
                    subprocess.Popen(["rm -r {}".format(photos)], shell=True)
                    print("borrado directorio \"{}\"").format(photos)
                    previous_state = 1
                elif current_state == 0 and previous_state == 1:
```

```
                    print "Sensor reading: "
                    previous_state = 0

            time.sleep(0.1)
    except KeyboardInterrupt:
        print "stopped by user"
        GPIO.cleanup()

#creates a new directory, with a given path
#if the directory exists, it won't create it
#   path: path where the new directory will be created
def mkdir_p(path):
    try:
        os.makedirs(path)
    except OSError as exc: # Python >2.5
        if exc.errno == errno.EEXIST and os.path.isdir(path):
            pass
        else: raise

#takes a video of the scene with the USB camera and return a Gfycat link containing the uploaded video
#the file is saved in avi format and the name is that of the current date (Y-m-d HH-MM-SS)
def take_video():
    print "taking video"
    currTime = datetime.datetime.now().strftime('%Y-%m-%d_%H-%M-%S')
    take_vid = subprocess.Popen(["avconv -f video4linux2 -s 480x320 -r 24 -i /dev/video0 -t 5 {}.avi".format(currTime)], shell=True)
    take_vid.wait()
    print "video taken"

    c = GfycatClient()
    response = c.upload_from_file("{}.avi".format(currTime))
    link = response["webmUrl"]

    print("video uploaded")
    return link

#posts a tweet with the photo taken
def alert_user(link):
    api.send_direct_message("hmzuleta",text="Motion has been detected. {}".format(link))
    print("tweet sent.")

#rings the buzzer
def make_sound():
    t=0
    while t<1:
        GPIO.output(buzzer, 1)
        GPIO.output(led, 1)
        time.sleep(1)
        GPIO.output(buzzer, 0)
        GPIO.output(led, 0)
        time.sleep(0.5)
        t+=1

def make_sound_thread():
    sound_thread = threading.Thread(target=make_sound)
    sound_thread.start()

#sets the "on" variable to True or False when the button is pressed
```

```
174.def check_button():
175.     global on
176.     while True:
177.         if GPIO.input(button):
178.             on = not(on)
179.             print "on = {}".format(on)
180.             time.sleep(0.5)
181.
182.def check_button_thread():
183.     button_thread = threading.Thread(target=check_button)
184.     button_thread.start()
185.
186.main()
```

Previa la ejecución del archivo debe verificarse que todas las librerías que se usarán en los `import` al comienzo del archivo estén instaladas.

Nótese que se ejecutan comandos de consola por medio de la definición `Popen` de la librería `subprocess`. Nótese también que el proceso de captura del video se lleva a cabo en la línea 141 por medio de un comando de la librería `Avconv` [6] al cual se le inyecta el nombre del archivo de video como una concatenación de la fecha y la hora actuales. El proceso de subir el video a la plataforma Gfycat se realiza en las líneas 145 y 146, en las cuales se hace uso del objeto `GfycatClient` [7].

El proceso de notificación se lleva a cabo enviando un mensaje directo de Twitter al usuario "hmzuleta" (*hardcoded* para el enfoque de este proyecto), lo cual ocurre en la línea 154.

Para realizar el monitoreo local de la cámara USB se utiliza la librería `motion`. Ésta permite utilizar la tarjeta Raspberry Pi a manera de servidor LAN para visualizar la habitación de manera medianamente remota. Se configuró para que la calidad no fuese demasiado alta y se minimizara la latencia. Notamos que entre más alta la resolución, la tarjeta de desarrollo se veía menos capaz de procesar todo el video, por lo que algunos cuadros del video eran saltados y perdía sentido el monitoreo si no era cercano a tiempo real. Se obtuvieron valores de latencia cercanos a 1 segundo, lo cual consideramos como aceptable, pues se alcanzan a detallar los rostros sin problema. Con el fin de que los servicios de monitoreo LAN y toma de video no entraran en conflicto, el primero se detiene una vez se detecta movimiento, liberando la cámara USB como recurso y se procede a grabar el video que será enviado al usuario. Para evitar problemas de latencia con la cámara, Raspberry Pi Foundation recomienda utilizar el módulo de cámara oficial que se conecta directamente a la tarjeta y, con base en nuestra experiencia, nosotros recomendamos el uso de versiones más moderna de la dicha tarjeta de desarrollo, como la Raspberry Pi 2 o la Raspberry Pi Zero.

Trabajo futuro

El principal punto de extensión del proyecto, en el cual consideramos hay más potencial, es el acoplamiento a la plataforma OpenMTC, de manera tal que el sistema se pueda comunicar con otros dispositivos en una red M2M. Un caso de uso importante tras dicho

acoplamiento sería la integración con el sistema de iluminación LED inteligente del recinto para mejorar la las condiciones de luz a la hora de tomar el video, mitigando así el actual punto de mejora que representa que el sensor se active cuando la habitación está a oscuras.

Sin embargo, estamos al tanto de que tanto la idea como su implementación se encuentran en etapa de gestación y que hay puntos de mejora básicos antes de que este sistema pueda llegar a un estado de producción en masa y comercialización. Algunos de estos aspectos a mejorar son: Configuración del servicio de monitoreo utilizando Cloud con el fin de permitir monitoreo fuera de la red local; aumento del número de sensores, ubicando al menos uno en cada habitación relevante del recinto; un medio para cambiar la cuenta de usuario de Twitter a la cual se desea que lleguen las notificaciones; autenticación ante el sistema por medio de servicios de localización del Smartphone del usuario, donde se detectaría utilizando herramientas de geolocalización que el usuario salió del lugar para poder activar el sistema o viceversa; finalmente, una adaptación para comercios, añadiendo a la actual solución ad-hoc varios tipos de sensores según sea la necesidad, v.g., si hay una bodega, enviar alarmas sobre la humedad y la temperatura, adicional a la alerta de un posible intruso, etc.

Bibliografía

[1] El Tiempo, «Ladrones, inseguridad y situación de robos en Colombia,» 9 Febrero 2015. [En línea]. Available: http://www.eltiempo.com/multimedia/especiales/ladrones-inseguridad-y-situacion-de-robos-en-colombia/14951881/1.

[2] El Tiempo, «Mas de 100 colombianos son atracados cada hora,» 26 Octubre 2014. [En línea]. Available: http://www.eltiempo.com/politica/justicia/delincuencia-comun-y-robo-a-viviendas-informe-del-dane/14743120.

[3] DANE, «Aumenta en 2,3 puntos porcentuales la proporción de hogares que poseía conexión a internet en 2014,» 2015 Abril 2015. [En línea]. Available: https://www.dane.gov.co/files/investigaciones/boletines/tic/cp_tic_2014.pdf.

[4] «Buy Canary's smart home security system,» [En línea]. Available: http://store.canary.is/.

[5] «Buy Piper nv,» [En línea]. Available: http://store.getpiper.com/products/piper-nv.

[6] «avconv video converter,» Ubuntu Manpage, [En línea]. Available: http://manpages.ubuntu.com/manpages/precise/man1/avconv.1.html. [Último acceso: 30 noviembre 2015].

[7] «gfycat 0.1.3,» Python Package Index, [En línea]. Available: https://pypi.python.org/pypi/gfycat. [Último acceso: 30 noviembre 2015].

APÉNDICES

Apéndice A – Código Fuente sin formato

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import datetime
import time
import os, errno
import tweepy
import threading
import subprocess
import base64
import json
import requests
from base64 import b64encode
from client import GfycatClient


GPIO.cleanup()

led = 4                          #on-board LED pin number. Edit depending on
your Raspberry Pi configuration
default_dist = 0          #normal distance to opposite wall
dist0 = 7                 #GPIO pin number for distance sensor (analog
ultrasound sensor)
buzzer = 23               #GPIO pin number for the buzzer
button = 24               #GPIO pin number for the button
on = True                 #boolean that indicates whether the system is armed
photos = "./webcam/"
currTime = 0              #represents the current Datetime object

#tweepy local variables
#twitter handle is RPipg (RaspberryPiproyectodegrado), pw:raspberrypi
#Access Level: Read, write, and direct messages
#keys and access tokens generated on September 27, 2015
#apps.twitter.com
apikey = "cuPArm9C54e7S6MCMHjE8xLBu"
apisecret = "oPPDyRFACPy0mdSL3DCrfhkryY7nc5KGZDY4Yes69pFazzQZVo"
accesstoken = "3799681336-N73XQxwTBskGelccSg2FfKO8A6Cw9APvjhBGbgu"
accesstokensecret = "sjXPUziK34vUPDNLrcxuftAzGX9MCi8iYUJr3bgujfojM"
auth = tweepy.OAuthHandler(apikey, apisecret)
auth.secure = True
auth.set_access_token(accesstoken, accesstokensecret)
api = tweepy.API(auth)

print "Started\n"

#setting up the GPIO ports
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
```

```
#setup for the on-board LED
GPIO.setup(led, GPIO.OUT)

#this signals the program will start
#turns on the on-board LED three times, and leaves it on the last time
GPIO.output(led, 1)
time.sleep(1)
GPIO.output(led, 0)
time.sleep(1)
GPIO.output(led, 1)
time.sleep(1)
GPIO.output(led, 0)

#setup for the distance sensors, readings in cm
GPIO.setup(dist0, GPIO.IN)

#setup for the button
GPIO.setup(button, GPIO.IN)#, pull_up_down = GPIO.PUD_DOWN)

#setup for the buzzer
GPIO.setup(buzzer, GPIO.OUT)

#starts the entire security monitoring script
def main():
	#This script does the following:
	#	-Start the live feed though LAN
	#	-If dist is reduced significantly,
	#		the alarm will be sent to the user, with a video of the
scene, and
	#		the board will make noise.
	#	-Take a video when the alarm is triggered. It will be saved
locally and
	#		sent to Gfycat.
	#	-Tweet the alert. Send a PM to the user with the link to the
Gfycat video
	try:
		global on
		check_button_thread()
		#start the local webcam server
		subprocess.Popen("sudo service motion start", shell=True)
		print "Started webcam server."
		#connect to server and load settings from User

		#create the folder where the pictures will be saved
		mkdir_p(photos)
		current_state = 0
		previous_state = 0

		print "waiting for sensor to settle"
		while GPIO.input(dist0) == 1:
			current_state = 0

		#start checking on the sensors indefinitely
		while True:
```

```
                    if on:
                           print "Sensor reading: "
                           current_state = GPIO.input(dist0)
                           if current_state == 1 and previous_state == 0:
#something approached the sensor
                                  print "!! Possible intruder. Taking
action...\n"
                                  make_sound_thread()
                                  #stop the webcam server daemon
                                  subprocess.check_call("sudo service motion
stop", shell=True)
                                  print "Stopped webcam server."
                                  link = take_video()
                                  #restarts the LAN webcam server
                                  subprocess.check_call("sudo service motion
start", shell=True)
                                  print "Restarted webcam server."
                                  alert_user(link)
                                  time.sleep(1)
                                  subprocess.Popen(["rm -r {}".format(photos)],
shell=True)
                                  print("borrado directorio
\"{}\"").format(photos)
                                  previous_state = 1
                           elif current_state == 0 and previous_state == 1:
                                  print "Sensor reading: "
                                  previous_state = 0

                    time.sleep(0.1)
      except KeyboardInterrupt:
             print "stopped by user"
             GPIO.cleanup()

#creates a new directory, with a given path
#if the directory exists, it won't create it
#     path: path where the new directory will be created
def mkdir_p(path):
    try:
        os.makedirs(path)
    except OSError as exc: # Python >2.5
        if exc.errno == errno.EEXIST and os.path.isdir(path):
            pass
        else: raise

#takes a video of the scene with the USB camera and return a Gfycat link
containing the uploaded video
#the file is saved in avi format and the name is that of the current date (Y-
m-d HH-MM-SS)
def take_video():
      print "taking video"
      currTime = datetime.datetime.now().strftime('%Y-%m-%d_%H-%M-%S')
      take_vid = subprocess.Popen(["avconv -f video4linux2 -s 480x320 -r 24 -i
/dev/video0 -t 5 {}.avi".format(currTime)], shell=True)
      take_vid.wait()
      print "video taken"
```

```
	c = GfycatClient()
	response = c.upload_from_file("{}.avi".format(currTime))
	link = response["webmUrl"]

	print("video uploaded")
	return link

#posts a tweet with the photo taken
def alert_user(link):
	api.send_direct_message("hmzuleta",text="Motion has been detected.
{}".format(link))
	print("tweet sent.")

#rings the buzzer
def make_sound():
	t=0
	while t<1:
		GPIO.output(buzzer, 1)
		GPIO.output(led, 1)
		time.sleep(1)
		GPIO.output(buzzer, 0)
		GPIO.output(led, 0)
		time.sleep(0.5)
		t+=1

def make_sound_thread():
	sound_thread = threading.Thread(target=make_sound)
	sound_thread.start()

#sets the "on" variable to True of False when the button is pressed
def check_button():
	global on
	while True:
		if GPIO.input(button):
			on = not(on)
			print "on = {}".format(on)
			time.sleep(0.5)

def check_button_thread():
	button_thread = threading.Thread(target=check_button)
	button_thread.start()

main()
```