Problem Set 2: Raspberry Pi Home Security

Noah Buchanan
Derek Yocum
CS 4363: Internet of Things Development
University of Arkansas - Fort Smith
Fall 2021

October 29, 2021

Documentation

There are a couple of notes to make about the application: We found out last minute that our humiture sensor may not be working so it was not including in the final product. The physical button is used as an emergency shutdown, it does not function to start the program and bring it to a clean stop, that's what the button on the web page is for. We also had a problem with separate pins not working for sensors other than GPI017 so we had to make some workarounds. The code would be much cleaner if we had figured out our solution to fixing this sooner but it still works as intended.

The web page is launched when the application is first started. The application is idle by default and will not do anything until you click the button. When the button is clicked our sensors start activating and is actively searching for an object inside the range of acceptable detection. Once something is detected and while it is still detected a notification will pop up in the alerts box after the refresh that happens every 2 seconds occurs and the buzzer will be going off while the object is still there beeping in intervals of 0.5 seconds. I don't have pictures of the configuration of the breadboard unfortunately because I did not see that they were requested until I had already left UAFS.

application.py

```
import RPi.GPIO as GPIO
import cherrypy
import time
import random
import threading
```

```
import os
from datetime import datetime

# ******************************
# Name: Noah Buchanan, Derek Yocum
# Problem Set: PS2
# Due Date: October 29, 2021
# ******************************

BtnPin = 11
TRIG   = 11
ECHO   = 12
ds18b20 = ''

running = False
alert = ''


class Page:

    @cherrypy.expose
    def index(self):
        global running
        if running:
            running = False
            self.destroy()
        return """<!DOCTYPE html>
                <html lang="en">
                <head>
                    <meta charset="UTF-8">
                    <title>Security System</title>
                    <link href="/static/css/styles.css" rel="stylesheet">
                </head>
                <body class = "bod">
                <div>
                <h1 class ="title"> Security System &trade; </h1>
                </div class = "title">
                <form method="get" action="/remoteStart">
                    <button class = "btn">Turn On</button>
                </form>
                <div class="temp">
                <h1> Current Tempature: <h1>
                Not currently on
                </div>
                <div class ="alert"><h1> ALERTS: <h1>
                """ + alert +"""
```

```
            </div>
            </body>
            </html>"""

@cherrypy.expose
def remoteStart(self):
    print(alert)
    global running
    if not running:
        self.setup()
        running = True
        t = threading.Thread(target=self.sensors)
        t.daemon
        t.start()
    temp = self.read()
    return """<!DOCTYPE html>
            <html lang="en">
            <head>
                <meta charset="UTF-8">
                <title>Security System</title>
                <script type = "text/javascript">
                    function autoRefreshPage()
                    {
                        window.location = window.location.href
                    }
                    setInterval('autoRefreshPage()', 1000);
                </script>
                <link href="/static/css/styles.css" rel="stylesheet">
            </head>

            <body class = "bod">
            <div class = "title">
            <h1 class ="title"> Security System &trade; </h1>
            </div>
            <form method="get" action="/index">
                <button class = "btn">Turn Off</button>
            </form>
            <div class="temp"><h1> Current Tempature: <h1>
            """ + str(temp) + """ </div> <div class = "alert"> <h1> ALERTS:
            </div>
            </body>
            </html>"""


def sensors(self):
    self.button()
```

```
    def setup(self):
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(13, GPIO.OUT)
        GPIO.output(13, GPIO.HIGH)
        GPIO.setup(ECHO, GPIO.IN)
        GPIO.setup(BtnPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# Set BtnPin's mode is input, and pull up to high level(3.3V)
        GPIO.add_event_detect(BtnPin, GPIO.BOTH, callback=self.detect, bouncetim
        global ds18b20
        for i in os.listdir('/sys/bus/w1/devices'):
                if i != 'w1_bus_master1':
                        ds18b20 = '28-01201f862d36'

    def distance(self):
        GPIO.setup(TRIG, GPIO.OUT)
        GPIO.output(TRIG, GPIO.HIGH)

        GPIO.output(TRIG, 0)
        time.sleep(0.000002)

        GPIO.output(TRIG, 1)
        time.sleep(0.00001)
        GPIO.output(TRIG, 0)

        while GPIO.input(ECHO) == 0:
                a = 0
        time1 = time.time()
        while GPIO.input(ECHO) == 1:
                a = 1
        time2 = time.time()
        during = time2 - time1
        return during * 340 / 2 * 100

    def button(self):
        global running
        while running:
            if GPIO.input(BtnPin)==0:
                break
            dis = self.distance()
            if(dis < 20):
                t = time.time()
                t = datetime.fromtimestamp(t)
                global alert
                alert += ('<p>Motion detected : ({dist} cm) [').format(dist = int
                alert += str(t)
```

```
                alert += ']</p>'
                GPIO.setup(TRIG, GPIO.OUT)
                GPIO.output(TRIG, GPIO.HIGH)
                self.beep(0.5)
            GPIO.setup(BtnPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
            print (int(dis),'cm')
            print ('')
            time.sleep(0.5)


    def on(self):
        GPIO.output(13, GPIO.LOW)

    def off(self):
        GPIO.output(13, GPIO.HIGH)

    def beep(self,x):
        self.on()
        time.sleep(x)
        self.off()
        time.sleep(x)

    def read(self):
#       global ds18b20
        location = '/sys/bus/w1/devices/' + ds18b20 + '/w1_slave'
        tfile = open(location)
        text = tfile.read()
        tfile.close()
        secondline = text.split("\n")[1]
        temperaturedata = secondline.split(" ")[9]
        temperature = float(temperaturedata[2:])
        temperature = temperature / 1000
        return temperature

    def destroy(self):
        GPIO.setup(TRIG, GPIO.OUT)
        GPIO.setup(ECHO, GPIO.OUT)
        GPIO.output(TRIG, GPIO.HIGH)
        GPIO.output(ECHO, GPIO.HIGH)
        GPIO.cleanup()                          # Release resource

    def detect(self,chn):
        pass
```

```
if __name__ == '__main__':
    conf = {
        '/': {
            'tools.staticdir.root': os.getcwd()
            },
        '/static': {
            'tools.staticdir.on': True,
            'tools.staticdir.dir': 'public'
            }
        }
    cherrypy.quickstart(Page(), '/', conf)
```

styles.css

```
.btn {
        background-color: NavajoWhite;
        border: 2px black;
border: 10px solid black;
        color: Black;
        padding: 15px 32px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 40px;
        margin: 200px 145px;
        cursor: pointer;
        border-radius: 7px;
width: 250px;
border-width: 10px;
border-collapse: collapse;
        }
.alert {
color: Black;
background-color: NavajoWhite;
border: 10px solid black;
border-color: black 10px;
text-align: center;
float:right;
font-size: 12px;
border-radius: 7px;
margin: -250px 550px;
width: 40%;
height: 60vh;
border-width: 10px;
```

```
}
.temp {
color: Black;
background-color: NavajoWhite;
border: 2px black;
border: 10px solid black;
text-align: center;
float:left
font-size: 12px;
border-radius: 7px;
margin: -150px 30px;
width: 25%;
height: 15vh;
border-width: 10px;
}

.bod {
background-color: DarkSlateGray;
}

.title {
width:100%;
text-align: center;
font-size: 70px;
}
```