



QUT College

Assessment Cover Sheet

Please make sure you include the following details in your submission:

Students

| Student name | Student number |
|----------------------|----------------|
| Chanyoung Kim (Chan) | N11225271 |
| Yeeun Lee (Evelyn) | N11355191 |

Class

| | |
|---------------------------|----------------|
| Tutor | Geoff Polzin |
| Tutorial Day, Time & Room | TUT1, KG P-220 |

Statement of Contribution (for students working in a pair)

| | |
|----------------------|---|
| Chanyoung Kim (Chan) | 47%, DHT(Temp & Hum) sensor part, combine two web-page by button, video |
| Yeeun Lee (Evelyn) | 53%, PIR sensor + webcam part, video, write report |

Project Objectives

- 'Smart Home'

With the Raspberry Pi 3 A+ model, the hardware, smart home system architecture is configured, designed, built, and executed. Utilizing Raspberry Pi remote access, understand the structure of the existing smart home system and run the program using sensors to connect it to the internet via web pages.

IoT enables everyday devices such as toothbrushes, humidifiers, cars, and machines to collect data and respond intelligently to their users. By integrating everyday 'things' with the internet, it enhances people's productivity and efficiency. Using IoT technology, we improved the efficiency and stability of our homes.

There are two main functions, it is possible to verify CCTV by detecting people passing by the front door of the house using the human body sensor, as well as by verifying the temperature and humidity inside the house using the temperature and humidity sensor. Finally, using the 'flask' web framework, we can see real-time video and indoor environments.

Review and Discussion of Technologies Used

1) Technology used & Reasons for using technology

Programming language - Python(Database - sqlite3), HTML, Javascript, css, Database (.db)

Python3 was chosen as the primary programming language because the web framework was used as flask. The Python web framework, flask, is easy to use and less dependent, allowing it to run seamlessly on scale (Deery, 2022). In addition, Python is efficient at creating web applications that can process data quickly, and it also uses them for machine learning and artificial intelligence activities.

We needed a data store to graph the temperature/humidity figures, so we used Database. We attempted to use MariaDB to retrieve data values and create tables, but there was an installation error, so we used the sqlite3 module in Python. To provide visual effects or events such as images, buttons, and graphs, three front-end programming languages were used.

Alternatives: PHP

PHP is a server-side script. Before downloading php, it runs on the web server first. It is easy to use because the program code is invisible and is not affected by the version and type of web browser.

Camera images processing system - Opencv

OpenCV is a real-time computer vision programming library. It was installed due to its emphasis on real-time image processing. It also helps computers recognize images like human eyes, so OpenCV can handle anything that can be captured with a camera, such as CCTV images/videos. So we used it to design a webcam program.

Alternatives: motion

When OpenCV was not installed, this was the method that we used. First, activate the camera by running **\$ sudo raspi-config** on the terminal. And then, run **\$ sudo apt-get install motion** to install motion program. After this, if set the start_motion_daemon item to 'yes' in motion.conf, motion is run and the video is streamed on the web page. ([http:// ip_address : port](http://ip_address:port))

IoT service (send e-mail) - IFTTT

It is a software platform named after the programming conditional "If this, then that." It connects apps and services to trigger one or more automations associated with apps, devices, and services (Martin, 2020). Therefore, we built an automated system that uses 'webhooks' to send notifications through G-mail.

Alternatives: Zapier, Elastic.io

Zapier is the best-known alternative tool for IFTTT. With this technology, people can combine apps and devices to trigger specific outcomes from certain events. It has powerful app integration abilities, and there are many tools to automate. Elastic.io is also a web-based solution for app and API automation. Convenient and flexible interface allow to integrate user's own connectors ("The Best IFTTT Alternatives to Try in 2021", 2021).

Web framework - Flask

\$ sudo apt install python3-flask

It is a Python-based web framework. It specializes in the development of simple websites and API servers. It includes various functions such as DB linkage, template-type standards, and code reuse, and it can be used lightly, making it easy to quickly create small applications. We chose a lightweight flask to represent video streaming or graphs (Deery, 2022).

Alternatives: Wix, Squarespace, Wordpress and Django, Apache

Wix and Squarespace are appropriate for a nice DIY hobby website, and creating casual websites. Wordpress can achieve both of its previous goals, as well as business goals such as integration with business applications, due to its extensive flexibility and scalability.

Django is also a server-side web framework consisting of Python. When a request comes to the web server, it is delivered to the Django. Django websolver takes the address (URL) of the web page, checks what to do, and then processes the information in the view function. View is where we can put the logic of the application("Django Introduction", n.d.).

```
from django.shortcuts import render

def post_list(request):

    return render(request, 'blog/post_list.html', {})
```

The Apache software used during our tutorial class can also be an alternative. It is web server that provides static HTML or images. It has the disadvantage of not being able to interact with other services, however its structure is simple and processing speed is fast.

2) Operation works

OpenCV : Composed of CV + MLL + HighGUI. Image processing and algorithms are included, as well as cv with movement and tracking functions, MLL for data analysis via statistical classification and clustering functions, and GUI for video image and input/output functions. An algorithm is the main component of opencv's image processing. The main algorithms used in this process are binaryzation, outline detection, pattern recognition, and machine learning. When combined with other libraries, such as NumPy, it recognize image patterns and functions and performs mathematical operations. In order to process images, we used Python ("Save Numpy Array as Image in Python", n.d.). In opencv, image files are stored on different objects depending on the language used. In the case of python, it is stored in the Numpy array, which is a library for scientific calculations and provides the functions necessary to process multidimensional arrays("Basics of NumPy Arrays", 2022). When storing image data through opencv, a scalar value or vector representing the color of a pixel is expressed in a two-dimensional array. Due to this, Numpy is able to perform operations on arrays and matrices of large capacity quickly.

Flask : The flask has the following structure.

```
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route('/')
5  def hello():
6      return 'Hello, World!'
7
8  if __name__ == '__main__':
9      app.run()
10
```

1> Import the flask module in the source to declare it available, and write Flask() in the app to make the app available as a global object. 2> The part

that says @ is called as decorator, which processes HTTP requests by passing the request factor to the route function of the object in the app. And the function in 3> is associated with the decorator URL. If we request '/' at the end, that is, we can access <http://127.0.0.1> without port number, then hello world! will be appear on the web.

3) * How relate to the technologies presented in lectures

In this project, two sensors and one camera are H/W components such as Raspberry pi. Furthermore, IoT (Internet of Things) can be implemented by combining the technologies discussed above. When all devices and applications are connected to the internet, all devices become more smarter. Specifically, the flask framework is associated with a web composed of HTML + HTTP + URLs. The html data, that is, as used in the cctv live streaming, exists as a static file except for the live video. However, in the motion.py, the web page of the web server exists dynamically by using values in the database("Web pages are **generated on the fly** by web server" e.g. from templates and database data). This is called a dynamic web page.

And the Python development language, which is an attribute of the S/W development environment, is essential for the execution of the program. Also opencv is a processing software. It is a programming library aimed at real-time computer vision that processes images or videos using algorithmic methods, and is a cross-platform that can be used in various operating systems. Cross-platform applications can run on more than one platform. The kind of software is called multi-platform software.

Design and Implementation

1) Design & Implementation

Part of cctv.py - Using the cv2 module, connect the usb webcam and save pictures. ref and frame are the 'capture.read()' of the current stream video. <capture and video recording button, showing the number of all cases.>

```
# If the video was not well received (ref is false)
if not ref:
    break
else:
    frame = Image.fromarray(frame)
    draw = ImageDraw.Draw(frame)
    # xy is where the text starts, text is the string to output,
    # font is the font, and fill is the color of the letter
    draw.text(xy=(10, 15), text="Smart Home CCTV "+nowDatetime, font=font, fill=(255, 255, 255))
    frame = np.array(frame)
    ref, buffer = cv2.imencode('.jpg', frame)
    # Copy the current screen to frame1
    frame1 = frame
    frame1 = buffer.tobytes()
    # If you are not currently in the recording state and start_record
    # is true (press the recording button)
    if start_record == True and is_record == False:
        # Make it recorded
        is_record = True
        start_record = False
        video = cv2.VideoWriter("record " + nowDatetime_path + ".avi", fourcc, 15, (frame1.shape[1], frame1.shape[0]))
    # If you press the recording button again while you're recording
    elif start_record and is_record == True:
        is_record = False
        start_record = False
        video.release()
    # If you press the capture button
    elif is_capture:
        is_capture = False
        cv2.imwrite("capture " + nowDatetime_path + ".png", frame1)
    # If it's currently recorded
    if is_record == True:
        # Save the current frame to a video
        video.write(frame1)
    # Pile up the picture files and wait for the call
    yield (b'--frame\r\n'
           b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')
```

Part of appDhtWebHist.py - Using the sqlite3 module, store the values of temperature and humidity caught by the DHT sensor in the database. Then, use the 'getLastData' function to display the most recent value on the web page. And, draw a plot_temp() graph as image through the 'getHistData' function. Humidity graph is done in the same way.

```
# Use sqlite3, to access database data from python file
import sqlite3
conn=sqlite3.connect('../sensorsData.db')
curs=conn.cursor()
# Retrieve LAST data from database
def getLastData():
    for row in curs.execute("SELECT * FROM DHT_data ORDER BY timestamp DESC LIMIT 1"):
        time = str(row[0])
        temp = row[1]
        hum = row[2]
    #conn.close()
    return time, temp, hum

# get(history)data from DHT sensor
def getHistData (numSamples):
    curs.execute("SELECT * FROM DHT_data ORDER BY timestamp DESC LIMIT "+str(numSamples))
    data = curs.fetchall()
    dates = []
    temps = []
    hums = []
    for row in reversed(data):
        dates.append(row[0])
        temps.append(row[1])
        hums.append(row[2])
    return dates, temps, hums

# Temperature graph /Humidity graph also same structure
@app.route('/plot/temp')
def plot_temp():
    times, temps, hums = getHistData(numSamples)
    ys = temps
    fig = Figure()
    axis = fig.add_subplot(1, 1, 1)
    axis.set_title("Temperature [°C]")
    axis.set_xlabel("Samples")
    axis.grid(True)
    xs = range(numSamples)
    axis.plot(xs, ys)
    canvas = FigureCanvas(fig)
    output = io.BytesIO()
    canvas.print_png(output)
    response = make_response(output.getvalue())
    response.mimetype = 'image/png'
    return response
```

> To combine the two python file as 'run.sh'

part of cctv.py - push_capture button/ Others push button has same structure.

```
# Runs when the capture button is pressed
@app.route('/push_capture')
def push_capture():
    # Calling is_capture as a global variable
    global is_capture
    # Makes is_capture true
    is_capture = True
    return redirect(url_for('index'))

# The portion to host a website and show it to an accessor
if __name__ == "__main__":
    # Host is currently the internal IP of Raspberry Pi, and port is set arbitrarily
    app.run(host="0.0.0.0", port = "5000")
```

motion.py

```
import requests
import RPi.GPIO as GPIO
import time

def motion_detected():
    t = time.localtime()
    print("%d:%d:%d Motion Detected" % (t.tm_hour, t.tm_min, t.tm_sec))
    requests.post('https://maker.ifttt.com/trigger/motion_detected/json/with/key')
    requests.get('http://127.0.0.1:5000/push_capture')

def motion_not_detected():
    t = time.localtime()
    print("%d:%d:%d No motion" % (t.tm_hour, t.tm_min, t.tm_sec))

GPIO.setmode(GPIO.BCM)
pin = 7
GPIO.setup(pin, GPIO.IN)
try:
    print ("Waiting for sensor to settle...")
    print ("PIR modul test (CTRL+C to exit)")
    time.sleep(1)
    print ("Detecting motion...")

    while True:
        if GPIO.input(pin):
            motion_detected()

        else:
            motion_not_detected()
            time.sleep(3)

except KeyboardInterrupt:
    GPIO.cleanup()
print ("Quit")
```

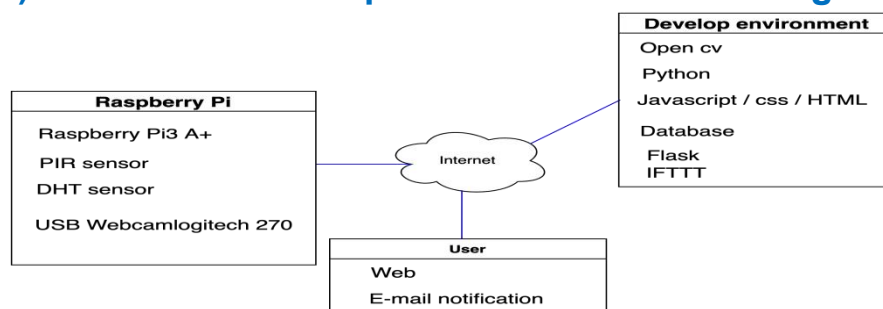
*motion.py -PIR sensor

GPIO: General purpose input/output interface, literally a general input/output interface. This allows physical external devices such as sensors to be connected and controlled.

Pin 7 means to plug the jumper wire connected to the sensor's High/Low output into GPIO07, No.26.

Data to be transmitted through the request module's 'post' method is put in the body of the HTTP message and transmitted. And, when the human detected, an e-mail is sent to the user using IFTTT. Also, through the 'get' method, two python files are connected by linking them with a button in cctv.py

2) S/W & H/W Development environment configuration



3) How it works with System Diagram

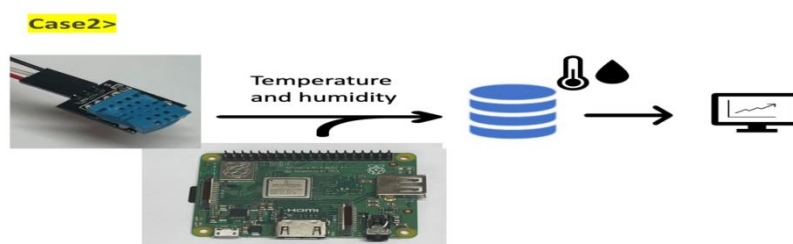
case1) CCTV streaming (PIR sensor + Webcam)

* Using the request module, connect two Python files to the address 'https://172.20.26.81:5000/push_capture/'. (push_capture is webcam function)



Using the PIR sensor connected to the Raspberry Pi, people's movements can be detected, as well as whether they are within or outside the range of the sensor(Li, 2019). A sensor such as this can be installed at the entrance of a house to allow people to view live video through a webcam camera. After a person is detected, an "motion_detected" email is sent to the email address specified in the 'motion.py', and the webcam screen is displayed. (two python file are combined as 'run.sh') The email was sent using the free IFTTT application. Specifically, when motion is detected, a streaming screen is captured by a capture button in 'cctv.py' which operates the webcam. Finally, the pictures are saved on the Raspberry pi.

case2) DHT sensor



Inside the house, data is collected using a DHT sensor that detects temperature and humidity. The sensor values are saved in 'logDHT.py', and the values of the saved file are saved in a database (SensorsData.db). Using Python's sqlite3 module, the stored data is loaded into a Python file. It also displays a graph on the web that displays a graph on the web that displays both temperature/humidity levels and historical records at the same time.

4) Problems faced during project

- **500 Internal Server Error:** The most difficult error to resolve. When a few errors were listed, it was difficult to find what the exact errors were.

- **jinja2.exceptions.TemplateNotFound(template):** Many errors occurred when creating a Python file and writing html inside the file due to the directory location.

- **Other challenging issues:** When ports combine two different pages into one button or like 'Package has no installation candidate', directory or file installation is difficult. Downloading 'Opencv', which has a video processing function sources, was difficult because there were a lot of downloads and it took a lot of time.

```
$wget -O opencv.zip https://github.com/opencv/opencv/archive/4.1.0.zip
```

```
$wget -O opencv_contrib.zip
```

```
https://github.com/opencv/opencv_contrib/archive/4.1.0.zip
```

& following for compiling

5) Experiments & Results


- Experiments

* Whether an e-mail is received using IFTTT when the human sensor detects motion. Using requests module, and post the url which trigger an event.

```

motion.py {} requests
1 import requests
2 import RPi.GPIO as GPIO
3 import time
4
5 def motion_detected():
6     t = time.localtime()
7     print("%d:%d:%d Motion Detected" % (t.tm_hour, t.tm_min, t.tm_sec))
8     requests.post('https://maker.ifttt.com/trigger/motion_detected/json/with/key/d5GSLr0giExPUXVg8H5achVIdnqLLL')
9     requests.get('http://127.0.0.1:5000/push_capture')
10

```



Webhooks via IFTTT action@ifttt.com


Motion Detect.

This mail is automatically sent when new movement is detected in your home by the Raspberry Pi sensor.

Please check the CCTV on the smart home web page.

* Does it run when press the 'capture' button on a web page? Green letters

Live Streaming



CAPTURE RECORD/STOP wait for record

```

leeyeeun ~ pi@raspberrypi: ~/102_mini_project/template/S...
[ WARN:0] global ./modules/videoio/src/cap_gstreamer.cpp (914) open OpenCV | GS
treamer warning: unable to start pipeline
[ WARN:0] global ./modules/videoio/src/cap_gstreamer.cpp (501) isPipelinePlayin
g OpenCV | GStreamer warning: GStreamer: pipeline have not been created
* Serving Flask app "smart_home_cctv" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployme
nt.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
172.20.27.157 - - [19/Jan/2023 00:16:47] "GET / HTTP/1.1" 200 -
172.20.27.157 - - [19/Jan/2023 00:16:48] "GET /video_feed HTTP/1.1" 200 -
172.20.27.157 - - [19/Jan/2023 00:16:55] "GET /push_record HTTP/1.1" 302 -
172.20.27.157 - - [19/Jan/2023 00:16:55] "GET / HTTP/1.1" 200 -
172.20.27.157 - - [19/Jan/2023 00:16:55] "GET /video_feed HTTP/1.1" 500 -
Error on request:
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/werkzeug/serving.py", line 323, in run_w
sgi
    execute(self.server.app)
  File "/usr/lib/python3/dist-packages/werkzeug/serving.py", line 314, in execut
e
    for data in application_iter:
  File "/usr/lib/python3/dist-packages/werkzeug/wsgi.py", line 506, in __next__
    return self._next()
  File "/usr/lib/python3/dist-packages/werkzeug/wrappers/base_response.py", line
45, in _iter_encoded
    for item in iterable:
  File "/home/pi/102_mini_project/template/Smart Home/smart_home_cctv.py", line
64, in gen_frames
    video.write(frame)
NameError: name 'video' is not defined
172.20.27.157 - - [19/Jan/2023 00:17:01] "GET /push_record HTTP/1.1" 302 -
172.20.27.157 - - [19/Jan/2023 00:17:01] "GET / HTTP/1.1" 200 -
172.20.27.157 - - [19/Jan/2023 00:17:02] "GET /video_feed HTTP/1.1" 200 -
172.20.27.157 - - [19/Jan/2023 00:17:06] "GET /push_capture HTTP/1.1" 302 -
172.20.27.157 - - [19/Jan/2023 00:17:06] "GET / HTTP/1.1" 200 -
172.20.27.157 - - [19/Jan/2023 00:17:06] "GET /video_feed HTTP/1.1" 200 -

```

* Using the button structure used by cctv.py, linked the streaming page to the temperature/humidity web page. Pursuing user convenience.

Diagram illustrating the capture-recapture process:

- Sequence of steps: TEMPT/HUM → CAPTURE → RECORD/STOP → wait for record → OCTV.
- Graph showing a step function (likely representing a signal or measurement) with a value of 30 over a time interval from 10 to 12.
- A blue arrow indicates the flow from the graph back to the TEMPT/HUM step, completing the cycle.



References

- Basics of NumPy Arrays. (2022, April 26). Geeksforgeeks.
<https://www.geeksforgeeks.org/basics-of-numpy-arrays/>
- Deery, M. (2022, August 1). What Is Flask and How Do Developers Use It? A Quick Guide. Careerfoundry. <https://careerfoundry.com/en/blog/web-development/what-is-flask/>
- Django Introduction. (n.d.). Mdn. <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>
- Li, I. (2019, August 3). PIR Sensor: Overview, Applications and Projects. Seeedstudio. <https://www.seeedstudio.com/blog/2019/08/03/pir-sensor-introduction-and-how-pir-motion-sensor-works-with-arduino-and-raspberry-pi/>
- Martin, J. A., & Finnegan, M. (2020). What Is IFTTT? How to Use If This, Then That Services. Computerworld.
<https://www.computerworld.com/article/3239304/what-is-ifttt-how-to-use-if-this-then-that-services.html>
- Save Numpy Array as Image in Python. (n.d.). Java2blog.
<https://java2blog.com/save-numpy-array-as-image-python/>
- The Best IFTTT Alternatives to Try in 2021. (2021). Ecommerce Booth.
<https://ecommercebooth.com/best-ifttt-alternatives/>