University of BRISTOL

DEPARTMENT OF COMPUTER SCIENCE

Energy footprinting of WordPress websites on the user side

Zhenni Qian

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering.

January 7, 2022

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

[illegible]

Zhenni Qian, January 7, 2022

Acknowledgements

I would like to thank my supervisor, Dr.Kerstin Eder, for her contribution to the project idea and supervision during this project. Furthermore, I acknowledge that my second supervisor, Maja Kirkeby, provided patient guidance and helpful resources on the direction of the project.

Ethics Statement

This project did not require ethical review, as determined by my supervisor, Dr.Kerstin Eder.

Executive Summary

Reducing carbon emissions is an issue that every nation is working on. In order to achieve the goal of decolonization, it is important to first understand how much carbon is being emitted by users. In other words, the first step towards carbon neutrality should be to measure the carbon footprint of the user side. The main purpose of this project is to explore the impact of different factors on the energy usage and carbon footprint of user interactions with WordPress-based websites.

In this project, automated testing of web pages will simulate user browsing pages, clicking buttons and downloading images, and the energy consumed during this process will be collected and monitored by external devices, including Raspberry Pi and other miscellaneous devices. The project focuses on exploring website size, website themes, browsers used and filters, all of which would lead to similarities and differences in energy consumption.

The main contributions of this paper are:

- Discovery of energy data from running Selenium automated tests in headless mode to simulate user interactions, together with carbon emissions of WordPress sites in different scenarios.
- Exploration of the impact of website size, free theme on WordPress, browser and filter on energy consumption. These factors have direct impacts on energy consumption, but the effects are independent of each other.
- Recommendation on energy-saving measures according to the analysis results, which can be used for web development or user browsing, such as browsing with the Edge browser and setting filters on sites.

Supporting Technologies

- I used WordPress to set up test-sites.
- I used Selenium 3.141.0 to build headless browser testing, together with chromedriver 96.0.4664.45, geckodriver 0.30.0 and edgedriver 96.0.1054.53.
- I used Raspberry Pi 3 and 32GB SD card supplied by my supervisor.
- I used AEOTEC smart switch 6 and Z-stick to collect and transfer energy data.

Contents

1 Contextual Background **1**
- 1.1 Global warming . . . 1
- 1.2 Motivation . . . 2
- 1.3 Outline . . . 3

2 Technical Background **6**
- 2.1 WordPress and themes . . . 6
- 2.2 Local energy footprint . . . 9
- 2.3 Energy monitoring system . . . 10

3 Project Execution **13**
- 3.1 Test tools . . . 13
  - 3.1.1 Selenium . . . 13
  - 3.1.2 Headless browser automation . . . 14
- 3.2 Raspberry Pi . . . 16
- 3.3 Zwave to MQTT . . . 17
- 3.4 Data collection . . . 18

4 Results and Discussion **20**
- 4.1 Web size . . . 20
- 4.2 Theme . . . 21
- 4.3 Filter . . . 22
- 4.4 Browser . . . 23

5 Conclusion **28**
- 5.1 Carbon Emissions . . . 28
- 5.2 Project Status . . . 28
- 5.3 Future . . . 29
  - 5.3.1 Testing directions . . . 29
  - 5.3.2 Website layout . . . 30
  - 5.3.3 Browser exploration . . . 30

List of Figures

1.1 A graph showing the annual changes in temperature of global surface as simulated by using human and natural factors. 2

2.1 The diagram from W3Techs indicates the historical trends in the usage of content management systems since January 2011 6
2.2 Layout of theme Mayland 7
2.3 Layout of theme Maywood 8
2.4 Layout of theme Rivington 8
2.5 Simplified representation of a client-server model, showing several elements on user side 9
2.6 MQTT publish-subscribe communication model 10

3.1 The set up in experiments used to collect energy consumption data from Raspberry Pi 18

4.1 A 3D model of energy consumption, showing the layers of energy usage with different website sizes 21
4.2 A graph showing the energy usage of downloading with different themes 22
4.3 A graph showing the energy usage difference between downloading on the page with and without filter 23
4.4 A graph showing energy consumption of downloading with three browsers 24
4.5 A graph showing energy consumption of downloading with Edge browser 24
4.6 A graph showing energy consumption of downloading with Chrome browser 25
4.7 A graph showing energy consumption of downloading with Firefox browser 25

Chapter 1

Contextual Background

1.1 Global warming

It is known to all that the world has been facing the challenge of the degradation of the natural environment and economic damages due to the reckless emission of carbon dioxide into the atmosphere[39]. Regarding the economic cost, the concept of the social cost of carbon is introduced to evaluate the damages by characterizing how extra carbon dioxide emissions influence future economic outcomes by changing the weather and temperature[14]. Similarly, this paper explains that[**1804**] release of only one ton of CO2 at present will increase the total amount of energy expenditures, approximately £3.87, in the future to a great extent.

The greenhouse effect, the root of global warming, is caused by the particular ability of carbon dioxide to absorb more heat than other gases. The concentration of carbon dioxide in the atmosphere acts as a regulator of the water-vapour content of the atmosphere and this gas plays a decisive role in helping the Earth to regulate the temperature. However, the previous papers have shown[19] that, an increase in atmospheric CO2 concentration only raises global temperatures slightly, suggesting that human activity is most likely to be responsible for global warming.

This year, 2021, is consistent with the long-term global warming upward trend of approximately 0.2°C every decade as was expected[37]. According to the latest IPCC Sixth Assessment Report published by the first of three working groups, the temperature of the global surface was 1.09°C higher in the decade between 2011-2020 than between 1850-1900[35], this would be regarded as a red flag for humanity. Notably, the report also confirms the statement of fact that the increase in carbon dioxide over the industrial era is an inevitable result of human activities[6].

The following chart in the report indicates the annual temperature changes of global surface. The light brown areas demonstrate the calculated warming caused by both human and natural factors. Besides, the areas in green show how the global surface temperature would have evolved if only natural causes were considered, for example, volcanic activities. It is apparent that the increase in temperature caused by human activities during the last decade is far greater than that caused by nature. This comparison of the simulations and observations makes it clear that humans are supposed to take main responsibility for climate change and global warming. The issue of global warming is affecting the living environment in every region, and some negative changes people have experienced are irreversible.

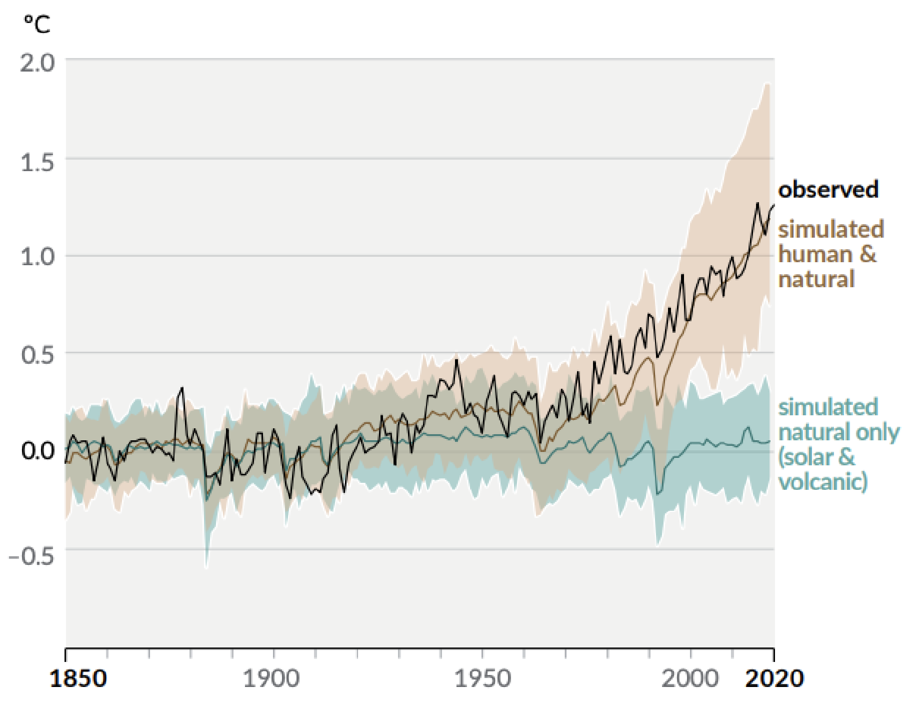


Figure 1.1: A graph showing the annual changes in temperature of global surface as simulated by using human and natural factors.

To achieve the long-term goal of cutting global emissions, nearly every nation all over the world signed up for the Paris climate agreement. This agreement is a landmark[32] in the multilateral climate change process, for the reason that it brings all countries into a common cause to undertake efforts to combat this global trend. According to the agreement[24], it aims to limit global temperatures to well below 2 this century and to pursue efforts to keep this figure under 1.5 degrees Celsius. In fact, the scientists forecast a net zero by the middle of this century if the global emissions could be cut in half by 2030, and we are able to reverse some of that temperature increase eventually. Therefore, reaching the target requires energy-saving strategies to work into our daily.

1.2 Motivation

By benefiting from the rapid improvements of the current sites and browsers, people are able to readily access website content and directly interact with the interface, such as downloading and saving files in local storage, by simply utilizing the browser application on their devices. In the context of this trend, user interactions are getting complicated and vary across different websites depending on the target audience, which can lead to the alteration of energy usage. A phenomenon in website power study has been observed: the energy consumed during the process of user interactions will not only influence the performance of website, but also affect the user viscosity. For example, Pinterest, an image sharing site, has improved the energy performance of the website by adjusting the layout of their pages, realizing a 40% reduction in perceived waiting time as well as 15% increase in both search engine traffic and sign-ups[28]. On the other hand, the concept of website sustainability raises the awareness about saving energy. In fact, it's a key element in inspiring green design aiming to shaping conscious energy consumption habits. Thus, users are motivated to visit the websites that are created to conserve energy[10].

Currently, information and communications technology account for between 5% and 9% of total power consumption, and the sustainable development suggests an action-oriented transformation of energy system, which is from smart networks to user management[9]. This energy usage of ICTs is broadly categorized into three components, they are data centers, transmission network and local client[5], among which the energy consumed by users accounts for the most significant proportion. However, the majority of current research into the origin of the energy generally centers on the factors of both transmission networks and data centers, which are related to data management and computing subsystems. For a more in-depth analysis of energy consumption on the user side, this project starts with the local energy footprint to explore the energy consumption caused by a specific interaction on WordPress-based websites. Users are expected to consume different levels of energy by scrolling pages, downloading images locally, and using web filters to access the target page directly on websites with different themes and sizes. These user-related factors are important components of local energy footprint and influence the total energy usage. By clarifying and comparing these factors, it is clear how user-related factors affect the local energy footprint. In the long term, it can inspire more strategies on sustainable web design and build energy-efficient websites to protect the environment.

1.3 Outline

The theme of the project is to explore the energy consumption of websites developed with WordPress, starting with a user's local energy footprint. I will set up user-side related factors as variables, collect the energy consumption data by running web automated tests, and compare the experimental results. This will then allow me to discover which site settings are conducive to reducing energy use, and which user selections achieve energy savings. The chapters of this project are arranged as follows:

1. Chapter 1 provides background information of the project and the motivation based on it.
2. Chapter 2 details the technical background of the project and defines the local energy footprint.
3. Chapter 3 explains the experimental procedure to explore the factors influence energy consumption and their difference.
4. Chapter 4 analyses and visualizes the results of the experiment.
5. Chapter 5, the final chapter, concludes the report and focuses on carbon emissions along with future improvements.

In summary, the purpose of this project is to evaluate the impact of four factors - site size, site theme, browser used and filters - on energy consumption on the user side. I will test with Selenium in headless mode on three popular browsers, Edge, Chrome and Firefox, to measure the energy consumed by a user downloading a specified number of images and transfer the data to the controller via a communication protocol. Exploring how these factors affect site performance, it is worth noting that this may not be linear, and I will then explain how energy usage can be mitigated through website settings. These specific objectives are therefore summarized as:

1. Run multiple sets of tests by using Selenium headless mode with different browsers, themes, sizes of WordPress sites as well as filter, and measure energy consumption on the user side.
2. Analyse whether these factors have an impact on user-side energy consumption, and how they affect the local energy footprinting.
3. Recommend web layout and browser selection strategies to achieve a reduction in user-side energy usage in sustainable web design.

Chapter 2

Technical Background

2.1 WordPress and themes

WordPress is an open-source content management system written in PHP that enables users to create, publish and store online content without charge.[36]. This software was originally designed as a blogging platform from 2003 but has established itself as a compatible content management system that support both traditional personal journal lists and advanced social media sites over the last few years and now has become the most popular web publishing platform in use. By October 2021, the current market share of the WordPress has increased to 42.8% (see 2.1)and the application has covered half of the top 10 million websites. It is already regarded as the most popular and best web design tools.

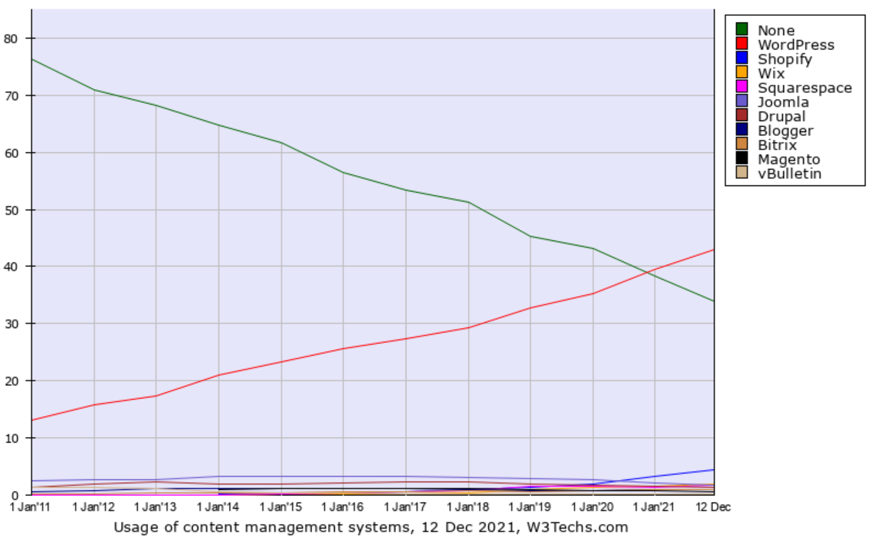


Figure 2.1: The diagram from W3Techs indicates the historical trends in the usage of content management systems since January 2011

One of the major advantages of WordPress[22] is a variety of features created by independent developers, including plugin modules and a template processor, referred to within this software as “Themes”. These additional features not only improve the functionality of the user interface but manage every element of site with respect to the creation and layout optimization by switching different themes. For users who confused in deciding themes to match the content, there exists a large-scale WordPress community organized to provide introduction and suggestion on their customized websites.

In addition to the visual communication in web design, the field of user interaction has witnessed the progress toward sustainability[11]. In particular, the web designers have called for green design to explore the significance of energy footprint minimization. By investigating the CSS style and HTML markup, as well as the functions for displaying content from open-source templates, the design of WordPress theme would be evaluated quantitatively[21]. The variations including number of elements, color contrast, font, and density in a theme could contribute to or detract from the performance and affect energy usage of the site. The rest of this section will describe the characteristics and content suitability of WordPress themes for testing.

Mayland

Mayland(Theme1) is a WordPress photography theme designed especially for commercial photographers, photojournalists as well as amateurs who expect to convert pictures to income. Mayland's specific tiled galleries set up with a slide-viewing presentation that highlights all the pictures on the homepage and allows visitors to enjoy an immersive online exhibition (see 2.2). Additionally, users may apply this theme on their online portfolio websites with the purpose of sell artworks to customers and fans. In this theme, text and images blocks are separated and the content occupied a large part of the page.

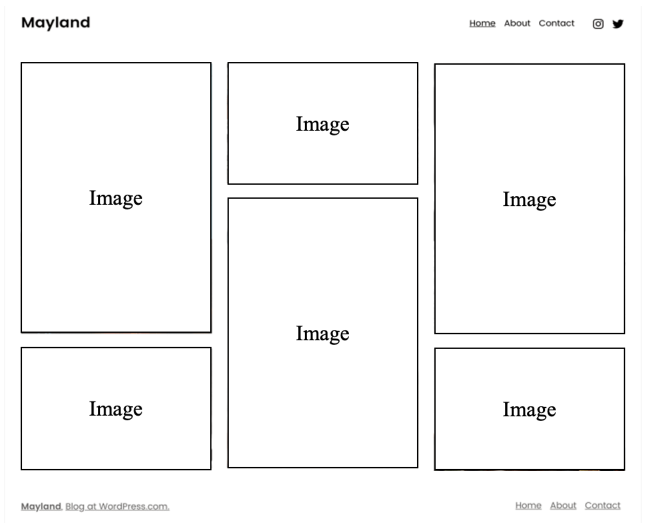
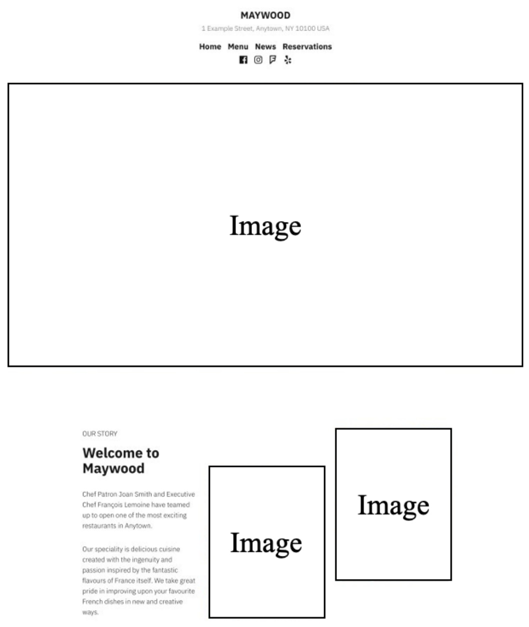


Figure 2.2: Layout of theme Mayland

Maywood

Maywood(Theme2) is a modern theme for restaurant that works for both formal dining establishments and coffee house. The layout of this theme features full-width imagery, along with a two-column page (see 2.3) to show visitors the combination of dishes images and text information in detail. Web designers are also able to add online ordering and table reservations with the plugin in the theme. This two-column layout requires a longer site length in website designs that displaying multiple images.


MAYWOOD
1 Example Street, Anytown, NY 10100 USA
Home Menu News Reservations
Image
OUR STORY
Welcome to Maywood
Image
Image

Chef Patron Joan Smith and Executive Chef François Lemoine have teamed up to open one of the most exciting restaurants in Anytown.

Our speciality is delicious cuisine created with the ingenuity and passion inspired by the fantastic flavours of France itself. We take great pride in improving upon your favourite French dishes in new and creative ways.

Figure 2.3: Layout of theme Maywood

Rivington

Rivington(Theme3) is designed as a WordPress template for realtors. This flexible theme highlights a recommended property that surrounded by other available projects (see 2.4). The feature of displaying images as many as possible on a single page improves the efficiency of visiting while extends loading time. Similar to Maywood, the block editor within this theme could customize square footage, and other details according to sites' needs.

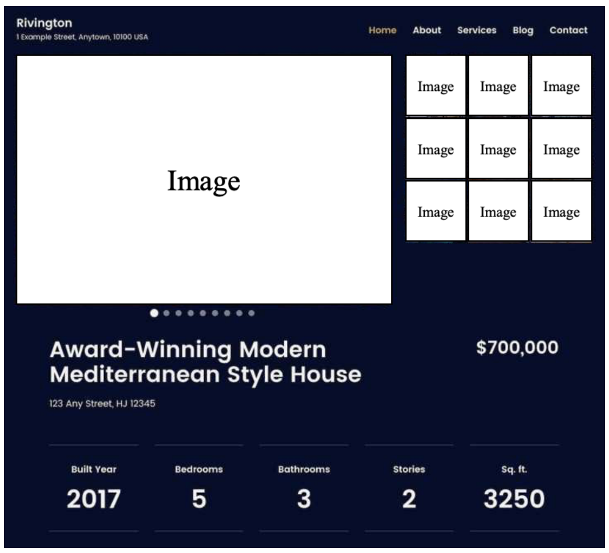


Figure 2.4: Layout of theme Rivington

2.2 Local energy footprint

The energy usage depends on both computer system and the properties of tasks selected to perform by users. Whether the computer is idle or not, it consumes energy all the time. On the other hand, those tasks determined by users contribute to additional energy consumption, therefore, the factors related to users are more flexible and important in controlling and reducing energy requirements.

In this project, local energy footprint refers to measuring energy based on a user-centered energy efficiency benchmark. In some cases, it might be measured by work completed after a fixed interval, in others, it might be measured simply in terms of units of specific work completed[40]. For example, I will record the energy consumption (and measure it in watts) during the period users scroll the page and download target images from browser to local (see 2.5). The total energy usage will vary with using difference browsers, applying filters and other components on the user side.

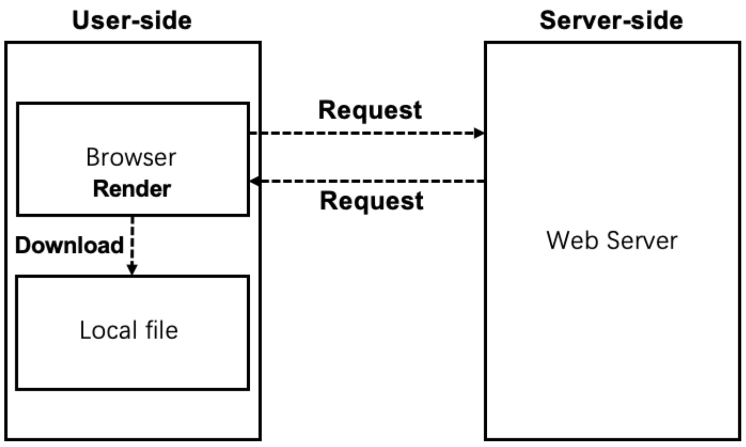


Figure 2.5: Simplified representation of a client-server model, showing several elements on user side

The ultimate goal of local energy consumption is to identify energy peaks and optimize energy system on client side by tracking user behavior and local settings. In order for the approach to perform effectively in most circumstances, it is supposed to capture and define the user activities that influenced by their habits.

In previous models[7], designers are able to test general user activities basically according to the default settings. However, building reliable models for explaining how each activity on the site could be carried out by every single user is hard to realize. The local energy footprint in this project, appropriate to compare the effects of web settings and user preferences, is focused on a common user activity——downloading images and save to local.

The energy footprints on user side are also indirectly influenced by the limitations of devices in use. The constraints of the devices, namely limited memory, certain browsers, and small size of screen[2], probably lead to more energy consumption while browsing. For example, less space on mobile device than desktop computer to display the data and images and hence the local energy footprints differ from each other. As suggested in the study from Steve Souders[30], only 9 percent of the overall page loading time is passed on the back-end and the rest 91 percent is spent on the front-end. It means that more energy would be consumed due to limited simultaneous connection and local latency issue.

The fundamental approaches to generate models for evaluating energy consumption are bottom-up and top-down. The “bottom-up” method collects energy consumption of each individual website component and analyzes the overall energy usage[4]. However, the ”top-down” method references more direct information on user interaction than the “bottom-up” [23], which makes the method incorporates overall energy consumption of user activities and thus provides more details on the user side. In order to consider both factors, the paper[17] adopts a new method for estimating energy, which named the unified method. In this method, the user preferences and local web settings are allowed to predict the energy for various scenarios on the user side.

2.3 Energy monitoring system

The data of energy consumption is transferred via Z-wave and MQTT within the monitoring system. Z-Wave is defined as[8] a standard wireless home automation protocol for controlling and monitoring lightweight appliances from a short distance. It uses extremely less power and runs on a mesh network that controlled by a certain device. As the controller for the Z-wave network[3], the AEOTEC Z-Stick is applied in monitoring the AEOTEC smart switch plugged into power supply and accessing energy data in the system subsequently.

This partial system is supported by Z-wave together with two devices in local network, the effectuation of data collection is depended on MQTT (Message Queuing Telemetry Transport) protocol[38]. This open protocol consists of one broker server and two types of clients including publisher and subscriber. According to the principle of the publish-subscribe communication model (see 2.6), components concerned with capturing certain information register their interest[16], meanwhile others process information do so by publishing their message. In this system, clients are allowed to subscribe to the topics they find interesting that are issued by publisher; MQTT broker plays a role of an intermediary for messages regarding the specific topic between publisher and subscriber.

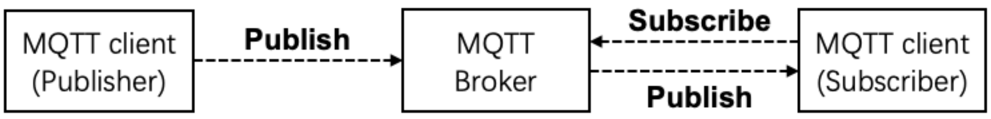


Figure 2.6: MQTT publish-subscribe communication model

In this project, by connecting smart switch(publisher) and the controller(subscriber) through the Z-stick(broker), the stream of energy data could be constantly transferred under one topic related to energy consumption. At this point, the data is stored on the Z-stick in a form that could be accessed by Z-wave. One approach of converting data into another form that can be accessed by MQTT protocol system is a secured z-wave to MQTT gateway, referred to as Zwavejs2mqtt[18]. Additionally, the clients need to import paho-MQTT client library into their code when obtaining real-time data on message topics as well as exporting for the following analysis.

Chapter 3

Project Execution

This chapter is about the execution of the experiment. The implement is based on several sizes of sites that built on WordPress and three free themes mentioned in the technical background chapter. These sets of websites will be tested on three different browsers(Edge, Chrome and Firefox) by using headless Selenium to simulate the behavior of users downloading images with the aim of monitoring and comparing the energy consumption on each scenario.

3.1 Test tools

3.1.1 Selenium

Selenium is an open-source umbrella project[27] that provides a series of functions geared to the requirements of automated testing for web-based applications. The main features of this tool are identifying web elements, simulating mouse or keyboard input, and comparing expected and actual results from the test web, etc. In addition to these flexible commands, Selenium is also allowed to execute the tests directly on popular web browsers, such as Firefox, IE, Chrome, Edge, etc[12]. For most web pages[15], it is manageable to find identifications for selected elements and extract their XPath queries by installing browser extension like XPath Helper. However, locating by XPath expressions will generally require more time to process information and contribute to longer-running tests in Selenium. Every element in a web page has a unique XPath, it means that XPath tools work with a higher accuracy in Selenium test. Furthermore, Selenium allows users to develop extensions by themselves, this feature makes it easy to design custom actions on the user side. Though some actions are JavaScript-restricted, such as downloading files from website, Selenium is still available for testing with several browsers.

The software also provides a domain language (Selenese), with which the tests could be processed in a number of programming languages. The previous paper [26] has explained that Selenium Python arranges API to write functional tests by applying Selenium Web Driver. The API, along with a suite of Selenium functions, allows the program to create interactions step by step with a page and evaluate the feedback of a test browser to different changes intuitively.

3.1.2 Headless browser automation

A headless browser is defined as[33] a web browser with no graphical user interface. That means the tests are executed but skip the step of loading the elements, and the graphical components are hidden from the web visitors. Since the pages aren't rendered in the test and bypass any interactions, this method for testing the performance of website is beneficial to manipulating the browser efficiently and effectively while integrating software delivery process with quality assurance[31].

Unlike a normal browser, a headless browser proceeds with each step of the test solely with command line interface or a Console. We are able to achieve headless mode tests by using Selenium WebDriver, a tool to automate site-based applications[29], it is used to drive and interact with browser applications natively as a user would do. Compared to Selenium RC, it does not require an inner engine but builds and operates test simply through a programming interface. In addition, the mode supports kinds of browsers including Mozilla Firefox, Google Chrome, Microsoft Edge, and Internet Explorer and speaks directly to the particular browser with the help of matched driver by executing Selenium WebDriver Script respectively.

A series of functions included in this project test such as clicking a filter button, downloading the images, and scrolling the page are operated by performing all the commands as per my script. Briefly, all user actions have corresponding Selenium code instructions. The sample script (written in Python) mentioned below is an example for running test script on WordPress website with Chrome webdriver. It will initially trigger Chrome browser for opening the designed website listed in the script and then locate every targeted element by their XPath. The entire script is aimed at stimulating the user to identify and save all the green images on the page and subsequently scroll it to the end. Because there is no GUI in headless mode, the action of downloading images by right-click with the cursor seems impossible to stimulate, the located images in headless mode are downloaded directly under this condition.

The WordPress-based sites in the script were customized in advance according to variables in the experiments, there are three test-sites:
https://theme1zhenniqian.wordpress.com/;
https://theme2zhenniqian.wordpress.com/;
https://theme3zhenniqian.wordpress.com/.

```
# Theme1 + Chrome + Without filter

import os # Imports the os package
import requests # Imports the requests package
from selenium import webdriver # Imports the selenium package
import time # Imports the time package

class Theme1ChromeWithout(object):
    def __init__(self):
        self.url = 'https://theme1zhenniqian.wordpress.com/'
        options = webdriver.ChromeOptions()
        options.add_argument('--headless') # set headless mode
        self.driver = webdriver.Chrome(options=options)
        self.driver.maximize_window()

        self.driver.implicitly_wait(20) # wait for 20s before throwing
                                                              an exception

    def __del__(self):
        self.driver.close() # close the browser after finishing tasks

    def run(self, n=0):
        headers = {
            "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_
            15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.
                                                  4664.55 Safari/537.36"}
        response = requests.get(url=self.url, headers=headers)
        response.encoding = 'utf-8'
        self.driver.get(self.url)

        js = "window.scrollTo(0,150)"
        self.driver.execute_script(js) # scroll to the first green
                                                                picture

        time.sleep(3)

        pic_list = self.driver.find_elements_by_xpath(
            "//img[contains(@src,
            'https://theme1zhenniqian.files.
             wordpress.com/2021/07/green.jpg')]") # find all the

        if not os.path.exists('./pics-1CW'):
            os.mkdir('./pics-1CW') # set a folder for downloaded pictures

        for i in pic_list:
            n += 1 # set a counter for picture list
            img_src = i.find_element_by_xpath("//img[contains(@src,
               'https://theme1zhenniqian.files.wordpress.com/2021/07
               /green.jpg')]").get_attribute('src')
            img_name = 'pic' + str(n) + '.jpg'
            img_data = requests.get(url=img_src,headers=headers).content
            img_path = 'pics-1CW/' + img_name

            with open(img_path, 'wb') as fp:
                fp.write(img_data)

            print(img_name, ' downloaded successfully!')

            if int(n) < len(pic_list)-1:
                self.driver.execute_script("arguments[0].scrollIntoView();",
                pic_list[int(n) + 1]) # If downloading is not done,scroll
```

```
                                                to the next green picture
            else:
                js = "window.scrollTo(0,document.body.scrollHeight)"
                self.driver.execute_script(js) # if downloading is
                            done, scroll to the bottom of the page

        time.sleep(2)

if __name__ == '__main__':
    theme1chromewithout = Theme1ChromeWithout()
    theme1chromewithout.run()
```

The following two lines of code are used to keep a track of user downloading images with the help of the filter. This user behaviour is represented in the script by the code that locates and clicks the filter button. I also collected the energy usage data during the period between opening the website and starting download, for the reason that the render time will change with size of website and hence influence the energy consumption on the user side. The energy consumption during this interval need to take into account in the final analysis.

```
filter_button = self.driver.find_element_by_xpath('//*[@id="post-2"]
                                                /div/div[1]/div[3]/a')
filter_button.click() # filter green pictures
```

3.2 Raspberry Pi

The major equipment of this experiment is Raspberry Pi, which is a microcomputer runs on the top of an ARM processor[20] and supports most operations as a normal computer. It could be connected to the Z-wave controller simply with the stick plugged into the USB port of Pi. It is important to note that the CPU(BCM2835) of Raspberry Pi is powerful but will not consume much energy, which means this device has little effect on testing results[25].

To carry out the Selenium automated tests mentioned in the above section, I set up the Raspberry Pi 3 in headless mode. In this case, I accessed the command line of the board remotely from my laptop with help of the Secure Shell (SSH) protocol which is a network communication protocol that enables two devices to communicate. More specifically, by finding the IP address of Raspberry Pi via Fing (a free network scanner application) and setting a SSH file on the boot partition of the SD Card[1], I'm able to connect the Pi to the exact wireless network where my computer in. Since the Raspberry Pi runs without hard drive, the test data will be eventually stored on a SD Card and accessed by the controller. This approach of connecting eliminates the requirements for additional auxiliary devices and obtain more accurate test data.

3.3 Zwave to MQTT

In this experiment, the Z-stick plugged into the Raspberry pi serves as the broker that allowing the smart switch connected to Z-wave net and data transfer between devices. According to the topic of the subscription, the targeted Z-wave energy data is stored on the Z-Stick, then I introduced the zwavejs2mqtt on the Pi in order to collect the data via MQTT in this scenario. The zwavejs2mqtt is a Z-Wave to MQTT gateway that has an ideal web interface for any Z-wave network related devices or node-specific commands. It provides an access to more detail information on the control panel UI by managing a port of the IP address of the Raspberry Pi.

The energy value changes every second on web interface and therefore I need to set a system on my computer for recording the data displayed and updated during a test. This system is applied for subscribing to the specific topic from Z-stick and getting energy data remotely from the topic. Additionally, a python client library named paho-mqtt together with Python code are required to realize the system.

The python script shown in this section is for connecting the computer to broker by listing IP address of Pi and printing the data of energy usage. I will exit the testing program and finish the data collection after downloading all the images on the website and the download progress would show on the terminal of Raspberry Pi, since the program prints a return statement every time a picture is successfully saved. In this project, the time of the experiment is also measured by calculating the quantity of energy data printed in each test and recorded as an energy measurement variable.

```
#MQTT demo for client

import paho.mqtt.client as mqtt

broker = "192.168.0.108" # This is the ip address of the raspberrypi

def on_connect(client, userdata, flags, rc):

if rc == 0:
print("Connected OK")
client.subscribe("zwave/D/50/0/value/66049")

else:
print("Could not connect to MQTT broker")

def on_message(client, userdata, message):
print(message.topic + ": " + str(message.payload))

client = mqtt.Client("raspberrypi")

client.on_connect = on_connect
client.on_message = on_message

print("Connecting to broker: ",broker)
```

```
client.connect(broker, 1883)

client.loop_forever() # Press Ctrl+C to exit after finishing download
```

3.4 Data collection

As discussed in the above sections and shown in the figure (see 3.1), the Raspberry Pi is physically connected with both smart switch and a Z-stick, and speak to controller via SSH. As a participant of the Z-wave net controlled by the stick, the smart plug also works as a power monitor measuring the power consumption of Pi. On the collecting stage, the data is transferred effectively from the stick to computer via MQTT.

I run the test code and MQTT code respectively in separate terminals of my computer and store the results in a csv file by inserting a command line on the controller. This makes it possible to manage the headless test on Pi and acquire every real-time data transferred by MQTT from my computer. Additionally, I calculated total energy usage of each test and write dataset in order to compare and model the results by running Pandas on Jupyter notebook.

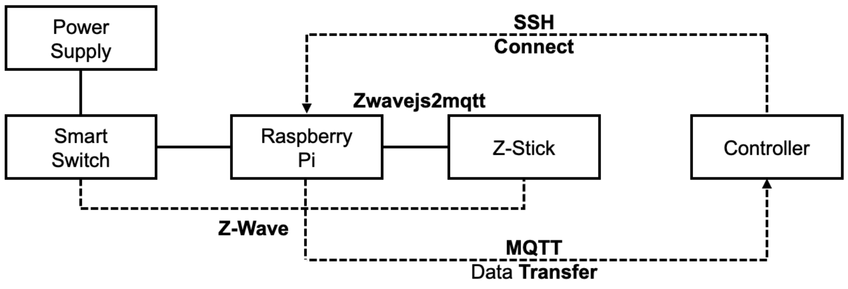


Figure 3.1: The set up in experiments used to collect energy consumption data from Raspberry Pi

Chapter 4

Results and Discussion

The results, along with an evaluation of the experimental approach are discussed and illustrated in the following sections. It is worth noting that the basic measurement of energy in data analysis is watt-hour (Wh), and energy expenditure of 1 Wh represents 3600 joules. For this, I calculated power consumption by multiplying the number of watts in each test by the number of hours it took. There are four variables in this project: the size of the web, the choice of the theme and browser, as well as whether the user visits with a filter or not. By controlling these variables above, we could learn how different user-side factors influence the energy usage of websites based on WordPress.

4.1 Web size

The size of the website varies with the number of images displayed on the site; this variable could be controlled by designing the page on the WordPress backend interface. In the default setting of the test site, the homepage is mainly divided into three parts: three filter buttons at the top, a certain number of fixed size (1641 bytes each unit) images in different colours, and 2000 bytes of lorem-ipsum text.

The majority of the experimental results without using a filter illustrate that the web size is directly proportional to the energy consumption on the user side, statistically, when 100 images are added on the web page, the average energy usage increases by approximately 0.05 Wh (see 4.1). The speed at which a page loads depends on the website size, which means that a reasonable compression of the site is effective in reducing load time and saving user energy. However, there is no significant difference in energy data between downloading 400 and 500 images on website designed with Theme1 by using Edge or Chrome. The reason for this situation is that once the website size exceeds a particular value (Approximately 1.86 MB in this experiment), the continuous increase in size will have minimal impact on the energy consumption under the case of these themes and browsers. The impacts of browsers and themes on site capacity and energy consumption will be discussed in the following sections.

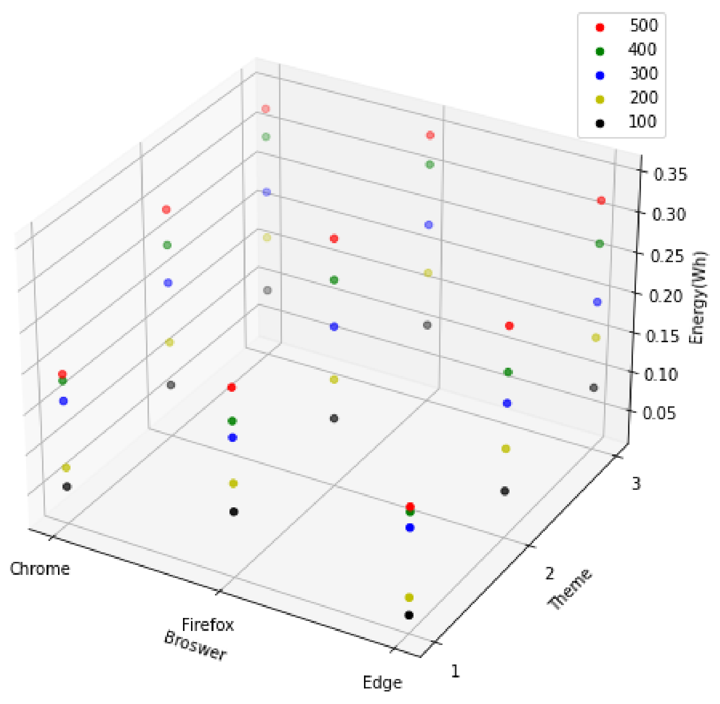


Figure 4.1: A 3D model of energy consumption, showing the layers of energy usage with different website sizes

4.2 Theme

In the technical background chapter, I discussed the characteristics of three free themes that are available for WordPress based sites. The analysis of the experimental results shows that the website theme is also a key factor that affects the energy consumption on the user side. In this test, I choose no filter and 500 targeted images as two controlled variables, because the more significant effects on energy value occur under these conditions. It can be seen from the graph (see 4.2) below that Theme 1 has the lowest average value at 0.19 Wh, the figure is much lower than Theme 2's energy consumption of 0.29 Wh, while Theme 3 consumes the most energy on average at 0.34 Wh. The difference in energy usage between the three sets of themes is comparable to 1-2 times additional energy every time adding 100 images on the page. The graph also shows that the three groups of browsers rank the same in terms of energy consumption by different themes, which also indicates that the browser is not directly related to the influence of the theme and that these energy differences can be explained by the characteristics of the website theme.

The number of colours and the size of the background image used in the theme templates could affect the rendering time of a web page, and websites with more colours and complex wallpapers lengthen the response time. Furthermore, by comparing the constructions of three themes, it is conspicuous that the multi-column layout mode consumes more energy than the image-text cross mode, this is due to the fact that the former mode enables users to find the aimed image faster, thus maintaining an image continuity on the page and ultimately completing the downloading task.

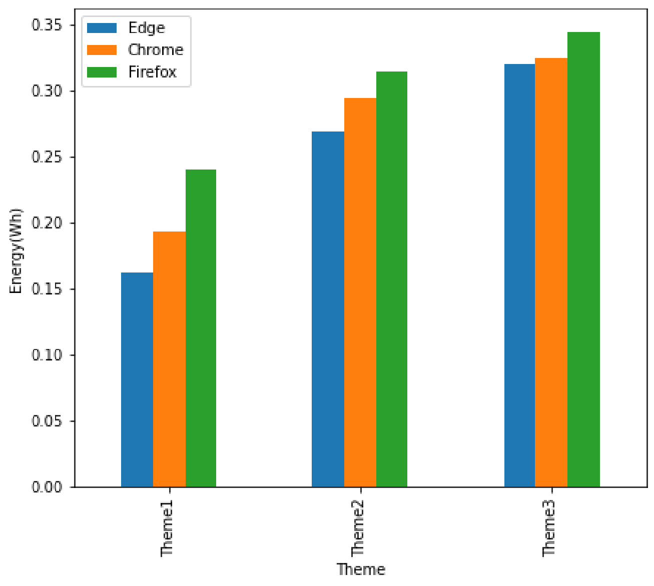


Figure 4.2: A graph showing the energy usage of downloading with different themes

4.3 Filter

In the experiment of downloading 500 green images, the test groups with the additional condition of using a filter appeared to have lower energy consumption on the user side. In terms of total energy usage for three sets of browsers(see ??), the filters saved 0.03Wh (Edge), 0.05Wh (Chrome) and 0.09Wh (Firefox) respectively. Overall, the energy consumption without using a filter is approximately 1.02 times higher than with a filter. Similarly, the energy consumption of downloading 100 images with a filter is 0.013Wh lower than the opposite, and hence this factor is not significantly related to the site theme or browser. During this experiment, the program is supposed to rapidly give a response and locate the target button according to the testing script, the energy usage of selecting a filter option is therefore ignored.

The approach of building filters is creating filter buttons that link to other sub-pages on the WordPress web design platform. The main purpose of the site filter is to give users the opportunity to remove everything that does not meet their requirements, so that remains what perfectly suits their expectations, which results in a more personalized and efficient browsing process.

Since the filter is designed at the top of the home page, users are allowed to click the filter button, skip loading the rest of the page and thus jump to the target page, this process would save user-side energy to a great extent. Note that the filtered sub-page presents fewer components than the main page, the energy for scrolling the page reduce in this way. An effective website filter is apparently an important element in energy savings, as even the basic operation of adding a filter on the page can achieve a growth of 26% in website conversion. Hick's Law[34], a law of psychology that applies to the web and interface design, also illustrates this point. It states that browsing a website without any filter options lead to two results: user will take longer time to select or they will exit immediately. It is clear that the time taken to find sections and the energy consumption on the user side can be saved by adding web filters.

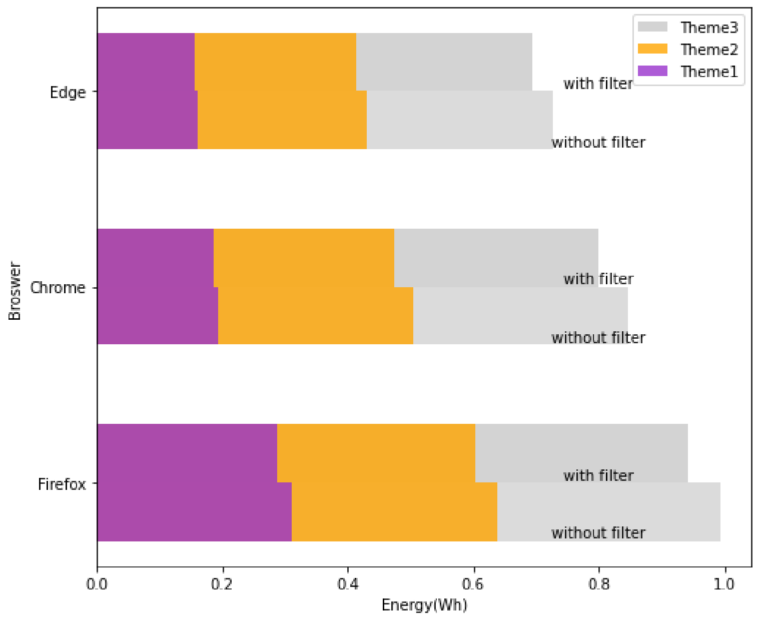


Figure 4.3: A graph showing the energy usage difference between downloading on the page with and without filter

4.4 Browser

This section will explain whether using different browsers could result in differences in energy consumption, and the comparison of energy usage between three browsers. As mentioned above, the two variables of website theme and filter have full independence from the choice of browser. Therefore, in this experiment, I collected the energy consumption data of users during the process of downloading between 100 and 500 images on a website designed with theme 1, without using any filters. Since the experiments were conducted in headless mode, I chose Chrome, Edge, and Firefox as three browsers that supported this test mode.

As can be seen from the graph below (see 4.4), there is a significant difference in energy consumption between the three browsers, regardless of the number of images downloaded. Firefox consistently ranks first in terms of total power consumption, with a value of 0.3 Wh for maximum images downloaded, which was 0.1Wh higher than Edge that makes the lowest power consumption, and this is the largest power difference in the parallel experiments. Such a hierarchical difference reflects the fact that, among the three sets of browsers, Edge is the first-rate energy saver, followed by Chrome and Firefox. The reason for this is that, as a simple browser, Edge has the additional feature that it does not have as many add-ons as Firefox or Chrome, thus it won't consume much energy in this test.

As the size of the site increases, the energy consumption rises gradually in each browser. When the site size reaches 300 images, the difference in user-side energy consumption between three browsers becomes apparent. The increase varies across the range of images. Specifically, as the site size increases to 400 images, the increase in energy consumption slows down from an average of 0.053Wh per 100 images to 0.043Wh per 100 images, and the browser energy gap stabilizes to a value as a result. Without other exceptions, we can expect the ratio of energy consumption among three browsers (Firefox: Chrome: Edge) to be closer to 1.5:1.2:1. In summary, a user's choice of browser also affects energy consumption, depending on the performance and settings of that browser in use.

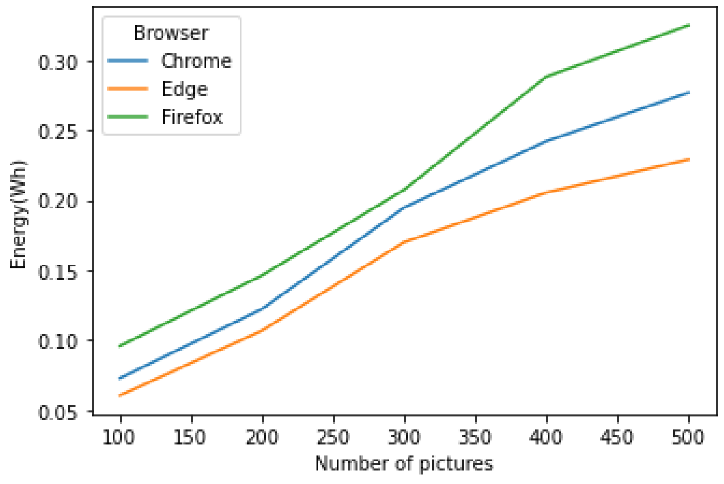


Figure 4.4: A graph showing energy consumption of downloading with three browsers

Compared to the other two browsers, Edge consumes the lowest total amount of energy on the user side, with the highest value even not exceeding 0.3 Wh. The following graph based on the energy consumption of Edge indicates that there is an overall upward trend in the value of energy consumption of loading and browsing sites with three different themes via Edge as the size increases, but there does not appear to be a linear relationship between browser and user energy. In the test of downloading 100 images, the average energy usage for the three themes was 0.06Wh per 100 images, and 0.065Wh per 100 images under a condition of downloading 500 images, it means that the growth rate of average energy consumption is relatively low when using Edge.

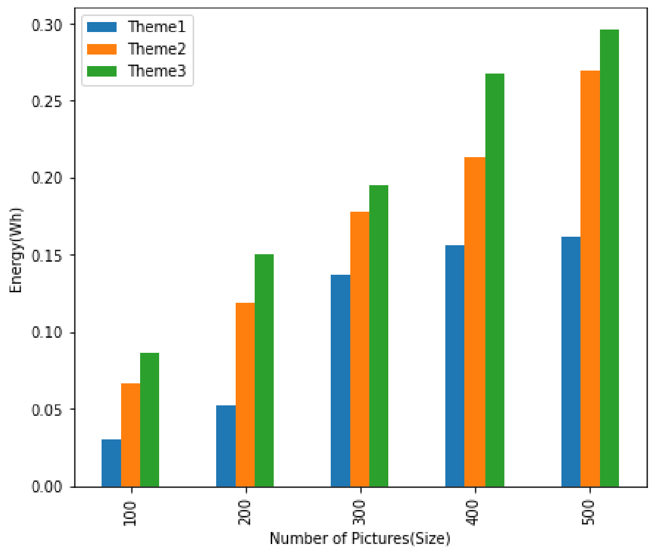


Figure 4.5: A graph showing energy consumption of downloading with Edge browser

Chrome is considered to be the most widely used browser, and the experimental results suggest that it's not the most energy efficient option. For Chrome, it takes at least 0.05Wh to download 100 images from WordPress sites designed with three different themes. However, its growth trend with site size is similar to that of Edge, the average growth rate of energy consumption when using Chrome is around 0.044 Wh per 100 images.

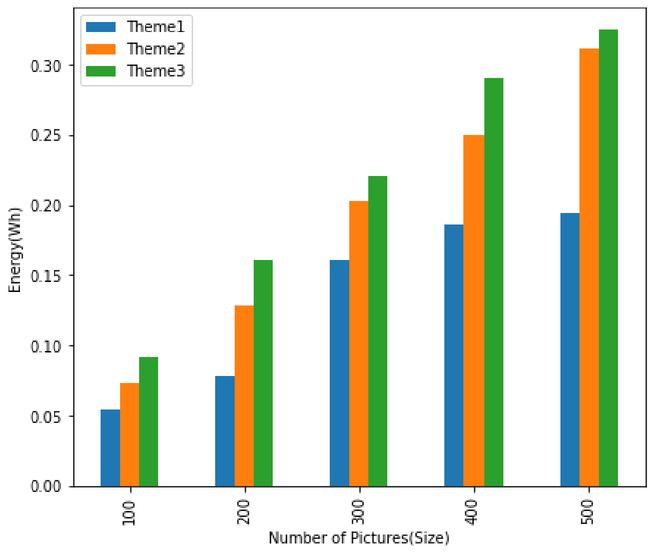


Figure 4.6: A graph showing energy consumption of downloading with Chrome browser

Regardless of site size and theme, Firefox consumes the most energy on the user side among three browsers. At the smallest site size, it has a minimum energy consumption of 0.085Wh, this value that starts to occur when downloading more than 200 images by Edge. Note that the average user energy consumption of the browser can reach 0.073Wh per 100 images. In addition, under the controlled condition of fixed number of images downloaded, the difference of energy usage from each website is subtle.

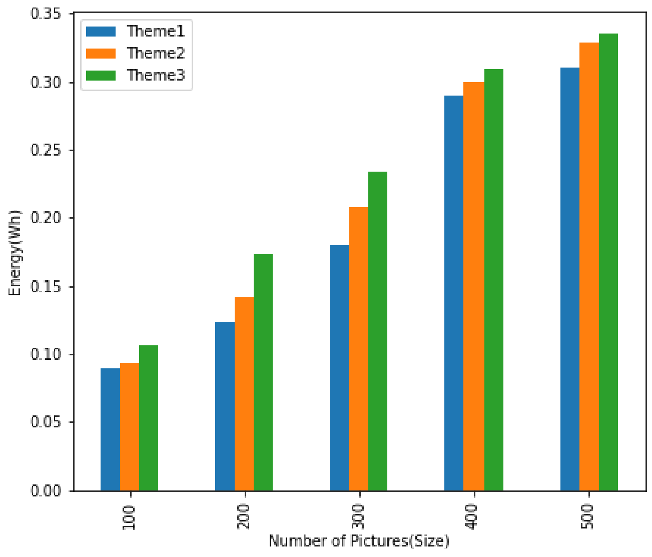


Figure 4.7: A graph showing energy consumption of downloading with Firefox browser

In this experiment, due to the limitations of the headless mode, only three browsers were selected for Selenium testing. In this case, the results of the experiment are based on the assumption that users choose one of the above three browsers for web browsing and downloading images.

Chapter 5

Conclusion

5.1 Carbon Emissions

When analysing the above result data, linking the energy consumed during user interactions to the CO2 emission figures allows for a more intuitive assessment of the impact of energy consumption. Based on the conversion factors provided in the UK Government's Greenhouse Gas Report: Conversion Factors (2021)[13], we are able to calculate and compare the carbon emissions from energy consumption on the user side, which makes the task of quantifying website emissions possible.

For the analysis, I selected the conversion factors of 0.21233 kg CO2 per kWh and 0.01860 kg CO2 per kWh for electricity generation as well as transmission and distribution respectively. The conversion factor of the whole process is thus approximately 0.00023098 kg CO2 per Wh. Combined with the results of experiments that measured different variables, in terms of website size, adding the equivalent of a million images to a website releases 1.155kg of carbon dioxide.

By calculating with the conversion factor of 0.15276 kg CO2 per kilometre for short flights in the table, this is equivalent to an average passenger travelling 7.56 km on short-haul flight. In addition, I estimate that downloading one million images without a filter would result in 1.5707kg of CO2 from Firefox, 1.2011kg from Google, and 0.9701kg from Edge. These CO2 emissions are equivalent to those emitted by small petrol cars driving 10.5572km, 8.0730km and 6.5203km respectively. Therefore, user-side energy consumption is influenced by site size and theme, browser, and filters.

5.2 Project Status

The current status of this project is that experiments and analyses have been completed based on the three objectives outlined in the first chapter.

As seen in throughout the project execution chapter, all energy data consumed during the process of simulating a user downloading the target image is collected by running multiple sets of Selenium tests in headless mode on Raspberry Pi. The control variables in the test include the theme and size of website, browser selected by user as well as the filter. The experiment is implemented to measure the amount of energy consumed on the user side in different scenarios.

The results and discussion chapter analyzes and explains the results extracted from the experiment, which fulfils the second objective. The energy usage results of browsing WordPress-based sites designed with different themes and sizes by using Edge, Firefox and Chrome respectively are presented in graphical form. The energy consumption chart is accompanied by the discussion of different factors on the energy consumption of the client and the analysis of the degree of influence on the energy usage. These factors can be correlated with user behavior to discover the potential causes of energy consumption under different conditions.

The discussion related to the prediction of carbon emissions and the possibility of saving user-side energy consumption are both completed in the conclusion chapter. The recommendations for website optimization are provided based on the results of the data analysis in the above two objectives.

Notably, in the evaluation and forecast of project, most of the external data, such as the conversion factors and the trend of carbon emissions are accessed in 2021. Although this could not be a typical year because of the changes in the current situation of the pandemic and development in the use of Internet technology, the impacts of user side factors can still be seen on most websites.

5.3 Future

5.3.1 Testing directions

Although the targeted dataset can be collected in this experiment, the accuracy of the data is restricted by some factors. The weaknesses of the experimental method have been enumerated and outlined in the project execution chapter. The main problems were caused by some of the shortcomings that present in the headless testing.

To begin with, I'm not able to measure the time required for loading the page separately, the reason being that the controller won't receive any return information from the program after the execution completed, and hence I could only calculate the time based on the duration between the start of the test and the moment when the first return from downloading successfully. However, this is apparently not the method of collecting data with highest accuracy.

In addition, due to the lack of GUI in headless mode, the test program could not simulate the complete operation of the user right-clicking and saving images during the downloading process. Instead, it saves the target image directly to local storage according to the XPath. This phenomenon not only affects the test time, but the data results will be underestimated. Because the cursor's movements consume energy in practice, although this power value might be subtle. In future projects that investigate energy consumption on the user side, we could collect more accurate data and results by applying monitor or software that supports web displays for testing.

In terms of filter settings, the buttons act as filters in this experiment with the assumption that users choose the filter option by clicking the button. This does not influence the result, but given that drop-down filters and input-box filters are more frequently used in website designs, so the impact of filter styles on user-side energy consumption is worth more study in future project.

5.3.2 Website layout

The layout of the page directly contributes to different user behaviour on a website. It is not difficult to see from the results of the experiment that designs with concise filters, block layouts and pages that use a smaller number of colours are more in line with green design standards. These design styles satisfy the user's requirements to rapid searching as well as shorter loading time, which could increase user retention and save energy on the client side at the same time.

5.3.3 Browser exploration

The differences in the impacts of the three browsers on user-side energy consumption, and the fact that Edge is the one with the lowest energy consumption has been explained in the results and discussion chapter.

It is clear that the effect of browser properties and settings upon energy consumption is a fascinating area of research, and these factors have significant implications for client-side energy savings. This extends to optimizing browser cache strategies, hiding unnecessary add-ons, and so on. The low-energy browser components could be applied without affecting the conversion rate of web pages.

Bibliography

[1] Kurniawan A. “Introduction to Raspberry Pi”. In: (2018), pp. 2–4. ISSN: 1574-1192. DOI: https://doi.org/10.1007/978-1-4842-4212-4_1.

[2] Eda Koksal Ahmed. “Transcoding web pages for energy saving on the client-side”. MA thesis. Middle East Technical University, 2016.

[3] Christopher W. Badenhop et al. “The Z-Wave routing protocol and its security implications”. In: *Computers Security* 68 (2017), pp. 112–129. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2017.04.004. URL: https://www.sciencedirect.com/science/article/pii/S0167404817300792.

[4] Ramon Bertran et al. “Counter-Based Power Modeling Methods: Top-Down vs. Bottom-Up”. In: *The Computer Journal* 56.2 (Aug. 2012), pp. 198–213. ISSN: 0010-4620. DOI: 10.1093/comjnl/bxs116. eprint: https://academic.oup.com/comjnl/article-pdf/56/2/198/1039131/bxs116.pdf. URL: https://doi.org/10.1093/comjnl/bxs116.

[5] Yi Cao et al. “Deconstructing the Energy Consumption of the Mobile Page Load”. In: *Proc. ACM Meas. Anal. Comput. Syst.* 1.1 (June 2017). DOI: 10.1145/3084443. URL: https://doi.org/10.1145/3084443.

[6] carbonbrief.org. *In-depth QA: The IPCC’s sixth assessment report on climate science.* [Online; accessed 21-December 2021]. 2021. URL: https://www.carbonbrief.org/in-depth-qa-the-ipccs-sixth-assessment-report-on-climate-science.

[7] Pietro Cottone et al. “User activity recognition for energy saving in smart homes”. In: *Pervasive and Mobile Computing* 16 (2015), pp. 156–170. ISSN: 1574-1192. DOI: https://doi.org/10.1016/j.pmcj.2014.08.006. URL: https://www.sciencedirect.com/science/article/pii/S1574119214001382.

[8] Salim Jibrin Danbatta and Asaf Varol. “Comparison of Zigbee, Z-Wave, Wi-Fi, and Bluetooth Wireless Technologies Used in Home Automation”. In: *2019 7th International Symposium on Digital Forensics and Security (ISDFS).* 2019, pp. 1–5. DOI: 10.1109/ISDFS.2019.8757472.

[9] enerdata.net. *Between 10 and 20% of electricity consumption from the ICT* sector in 2030?* [Online; accessed 23-December 2021]. 2018. URL: https://www.enerdata.net/publications/executive-briefing/between-10-and-20-electricity-consumption-ict-sector-2030.html.

[10] Earl Friedberg and Edward Lank. “Learning from Green Designers: Green Design as Discursive Practice”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. San Jose, California, USA: Association for Computing Machinery, 2016, pp. 1312–1323. ISBN: 9781450333627. DOI: 10.1145/2858036.2858124. URL: https://doi.org/10.1145/2858036.2858124.

[11] Earl Friedberg and Edward Lank. “Learning from Green Designers: Green Design as Discursive Practice”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. San Jose, California, USA: Association for Computing Machinery, 2016, pp. 1312–1323. ISBN: 9781450333627. DOI: 10.1145/2858036.2858124. URL: https://doi.org/10.1145/2858036.2858124.

[12] Nisha Gogna. “Study of Browser Based Automated Test Tools WATIR and Selenium”. In: 4.4 (2014), pp. 337–339. DOI: 10.7763/IJIET.2014.V4.425.

[13] gov.uk. *Greenhouse gas reporting: conversion factors 2020*. [Online; accessed 23-December 2021]. 2021. URL: https://www.gov.uk/government/publications/greenhouse-gas-reporting-conversion-factors-2020.

[14] M Greenstone, E Kopits, and A Wolverton. *Developing a social cost of carbon for US regulatory analysis: a methodology and interpretation. Review of Environmental Economics and Policy, 7 (1), 23-46*. 2013.

[15] A. Holmes and M. Kellogg. “Automating functional tests using Selenium”. In: *AGILE 2006 (AGILE’06)*. 2006, 6 pp.–275. DOI: 10.1109/AGILE.2006.19.

[16] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. “MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks”. In: *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE ’08)*. 2008, pp. 791–798. DOI: 10.1109/COMSWA.2008.4554519.

[17] Kiyo Ishii et al. “Unifying Top-Down and Bottom-Up Approaches to Evaluate Network Energy Consumption”. In: *J. Lightwave Technol.* 33.21 (Nov. 2015), pp. 4395–4405. URL: http://www.osapublishing.org/jlt/abstract.cfm?URI=jlt-33-21-4395.

[18] Daniel Lando. *zwavejs2mqtt*. [Online; accessed 9-December 2021]. 2022. URL: https://github.com/zwave-js/zwavejs2mqtt.

[19] Trevor M. Letcher. “1 - Why do we have global warming?” In: *Managing Global Warming*. Ed. by Trevor M. Letcher. Academic Press, 2019, pp. 3–15. ISBN: 978-0-12-814104-5. DOI: https://doi.org/10.1016/B978-0-12-814104-5.00001-6. URL: https://www.sciencedirect.com/science/article/pii/B9780128141045000016.

[20] Shawn Wallace Matt Richardson. *Getting Started with Raspberry Pi*. Marker Media, 2013.

[21] Derek Ohanesian. “Automated Evaluation of WordPress Theme Design”. In: (2014).

[22] Savan K Patel, V.R. Rathod, and Satyen Parikh. "Joomla, Drupal and WordPress - a statistical comparison of open source CMS". In: *3rd International Conference on Trendz in Information Sciences Computing (TISC2011)*. 2011, pp. 182–187. DOI: `10.1109/TISC.2011.6169111`.

[23] Yeager Vogt PE, CEM, and MBA. "Top–down Energy Modeling". In: *Strategic Planning for Energy and the Environment* 22.4 (2003), pp. 64–79. DOI: `10.1080/10485230309509626`. eprint: `https://doi.org/10.1080/10485230309509626`. URL: `https://doi.org/10.1080/10485230309509626`.

[24] Annalisa Savaresi. "The Paris Agreement: a new beginning?" In: *Journal of Energy & Natural Resources Law* 34.1 (2016), pp. 16–26. DOI: `10.1080/02646811.2016.1133983`. eprint: `https://doi.org/10.1080/02646811.2016.1133983`. URL: `https://doi.org/10.1080/02646811.2016.1133983`.

[25] Maik Schmidt. *Raspberry Pi: a quick-start guide*. Pragmatic Bookshelf, 2014.

[26] selenium-python. *selenium-python*. [Online; accessed 11-December 2021]. 2022. URL: `https://selenium-python.readthedocs.io/installation.html#introduction`.

[27] selenium.dev. *The Selenium Browser Automation Project*. [Online; accessed 11-December 2021]. 2022. URL: `https://www.selenium.dev/`.

[28] Junaid Shuja et al. "Survey of Techniques and Architectures for Designing Energy-Efficient Data Centers". In: *IEEE Systems Journal* 10.2 (2016), pp. 507–519. DOI: `10.1109/JSYST.2014.2315823`.

[29] softwaretestingmaterial. *Headless Browser Testing using Selenium WebDriver*. [Online; accessed 11-December 2021]. 2020. URL: `https://www.softwaretestingmaterial.com/headless-browser-testing-using-selenium-webdriver/`.

[30] Steve Souders. "HIGH PERFORMANCE WEB SITES". In: (2008).

[31] telerik. *What is Headless Browser Testing, When (and Why) to Use It*. [Online; accessed 11-December 2021]. 2021. URL: `https://www.telerik.com/blogs/what-is-headless-browser-testing-when-and-why-use-it`.

[32] unfccc.int. *The Paris Agreement*. [Online; accessed 23-December 2021]. 2021. URL: `https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement`.

[33] wikipedia.org. *Headless$_b$rowser*. [Online; accessed 11-December 2021]. 2022. URL: `https://en.wikipedia.org/wiki/Headless_browser`.

[34] wikipedia.org. *Hick's law*. [Online; accessed 22-December 2021]. 2021. URL: `https://en.wikipedia.org/wiki/Hick%5C%27s_law`.

[35] wikipedia.org. *IPCC Sixth Assessment Report*. [Online; accessed 22-December 2021]. 2021. URL: `https://en.wikipedia.org/wiki/IPCC_Sixth_Assessment_Report`.

[36] WordPress.org. *WordPress*. [Online; accessed 10-December 2021]. 2020. URL: `https://wordpress.org/about/requirements/`.

[37] yaleclimateconnections.org. *2021 was a remarkable year for Earth's climate*. [Online; accessed 21-December 2021]. 2021. URL: `https://yaleclimateconnections.org/2021/12/2021-was-a-remarkable-year-for-earths-climate/`.

[38] Ding Yi et al. "Design and implementation of mobile health monitoring system based on MQTT protocol". In: *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. 2016, pp. 1679–1682. DOI: 10.1109/IMCEC.2016.7867503.

[39] Kelvin O. Yoro and Michael O. Daramola. "Chapter 1 - CO2 emission sources, greenhouse gases, and the global warming effect". In: *Advances in Carbon Capture*. Ed. by Mohammad Reza Rahimpour, Mohammad Farsi, and Mohammad Amin Makarem. Woodhead Publishing, 2020, pp. 3–28. ISBN: 978-0-12-819657-1. DOI: https://doi.org/10.1016/B978-0-12-819657-1.00001-3. URL: https://www.sciencedirect.com/science/article/pii/B9780128196571000013.

[40] Chenlei Zhang, Abram Hindle, and Daniel M. German. "The Impact of User Choice on Energy Consumption". In: *IEEE Software* 31.3 (2014), pp. 69–75. DOI: 10.1109/MS.2014.27.