Streamlining Computer Maintenance with Automation Scripts

Chase Twyman

Kennesaw Mountain High School

Advanced Scientific Research IV

Amanda Dennis

December 1, 2024

Abstract

Computer repair technicians spend significant time conducting repetitive software computer maintenance tasks. Maintenance tasks technicians conduct include applying available updates, installing software, optimizing hardware and software, and creating a documentation report. A script can be developed to automate repetitive software maintenance tasks. This study uses the engineering and design process to develop and test an automation script on a Raspberry Pi Zero 2 W that a computer technician can use on a repaired computer. This study tests the automation script using quantitative variables such as duration of maintenance and number of completed tasks. I configured the Raspberry Pi Zero 2 W to emulate mouse and keyboard outputs through its micro-USB ports and to accept screen input from a Windows computer using an HDMI to CSI adapter. Using Microsoft Visual Studio Code, I developed a data-driven automated maintenance script in Python that is installed on the Raspberry Pi. I tested the automated Raspberry Pi device on eight Windows 10 and 11 laptops and desktops. The device had an average computer maintenance duration of 38 minutes and 13.5 seconds, with an accuracy of 85%. The automated maintenance script significantly reduced the time computer technicians spend performing maintenance in their eight-hour workday by 16% when performed on at least two computers. In conclusion, implementing this study's automated Raspberry Pi can help streamline computer technicians' workflow, increase workplace productivity, and save the resources needed for computer maintenance. I recommend future studies to test automated maintenance scripts using different devices on other types of computers.

Acknowledgements

I would like to thank my research and internship teachers, Ms. Dennis and Dr. Hunt, and I would like to thank my True-IT Inc. internship mentor, Cathy Stump. This research would not have been possible without them.

Table of Contents

Streamlining Computer Maintenance with Automation Scripts

Chapter 1: Introduction

According to Mearian (2022), a Windows computer update can take up to six hours to finish due to the large size of these updates. Although most updates take less than an hour, devices that are infrequently used require a consistent connection to install these extensive updates (Mearian, 2022). This explains why when updates first become available, many individuals are compelled to delay important updates to reduce workflow interruptions (Rajivan et al., 2020). While computer updates can be inconvenient for users, uninstalled updates pose more of a significant challenge for computer repair offices (Pourhadi, 2019). Technicians in computer repair offices are responsible for optimizing and keeping computers updated to improve the computer's lifespan, performance, and security (Bika, 2022; Huculak, 2022; Pourhadi, 2019). Due to these upkeep expectations, computer technicians spend a significant amount of time installing necessary updates and applying optimizations on computers, which delays awaiting workplace tasks and decreases workflow efficiency (Mearian, 2022; Pourhadi, 2019; Wingate, 2016). The lack of cost-effective tools makes automating the computer optimization and update process impractical for computer technicians (Pourhadi, 2019). Additionally, the documentation process of computer maintenance can often be tedious (Wingate, 2016). According to Wingate (2016) and Hanna et al. (2014), computer technicians must document who conducted the maintenance, the maintenance activities performed, the time the maintenance was performed, and the verification of completed maintenance tasks.

Currently, computer technicians perform numerous maintenance tasks manually (Bika, 2022). The most common maintenance tasks performed manually are applying updates, installing software, optimizing systems, and keeping documentation of applied maintenance

(Andersen & MoldStud Research Team, 2024; Bika, 2022; Hanna et al., 2014; Pourhadi, 2019; Urbitsch, 2023; Wingate, 2016). Common optimizations that computer technicians can apply to computers are removing temporary files and defragmenting hard drives (Huculak, 2022; Urbitsch, 2023). For the documentation process, computer technicians document who conducted the maintenance, what time the maintenance was conducted, which maintenance tasks were completed successfully and unsuccessfully, and hardware and software specifications (Andersen & MoldStud Research Team, 2024; Hanna et al., 2014; Wingate, 2016).

Statement of the Problem

The main issue is that computer maintenance technicians spend excessive time conducting repetitive maintenance tasks instead of performing tasks that require refined physical and cognitive skills (Pourhadi, 2019; Shin et al., 2024). Repetitive maintenance tasks include applying Windows updates, software installation, software optimization, and documenting hardware and software information (Andersen & MoldStud Research Team, 2024; Bika, 2022; Pourhadi, 2019; Wingate, 2016). The persistence of these manual tasks will lead to decreased workplace productivity and reduced time per shift (Pourhadi, 2019). Currently, computer technicians can automate maintenance tasks and documentation by writing scripts (Pourhadi, 2019). However, running scripts for computer maintenance and documentation has some drawbacks. First, the computer must be connected to a monitor, keyboard, and mouse for script installation, which uses valuable time and desk space. Additionally, after the computer is rebooted after an update, the technician must re-enter the computer's credentials to continue the script's execution. Furthermore, if the computer requires an operating system reinstallation, any saved scripts that could be executing or documenting the maintenance process on the computer will be lost during that reinstallation process. Instead, computer technicians need a lightweight, non-invasive, external tool that can automate maintenance tasks and documentation without taking up significant resources (mouse, keyboard, and monitor) and desk space.

Purpose of the Study

The purpose of this research is to develop an automation script for a Raspberry Pi Zero 2 W (RPZ2W) to execute maintenance tasks on computers in order to reduce the time that computer technicians spend on maintenance tasks and documentation. This research is important because it will provide a cost-effective tool that automates the computer maintenance process without using a physical mouse, keyboard, or monitor to complete tasks. The outcomes will be significant to computer technicians in computer repair companies by providing a product that will leave more time for other repair and maintenance tasks that require more cognitive and physical skills that cannot be automated (Shin et al., 2024). The goal is to reduce the time computer technicians must spend on computer maintenance tasks by 10% by automating repetitive tasks, such as checking for updates, collecting device hardware and software information, and installing necessary software (such as antivirus if not present). This automation will benefit the computer technician and the business they are working under due to the time it saves. This study will program and analyze the automated RPZ2W's impact on improving workplace performance by considering accuracy, time saved, and number of completed tasks. Usually, before a computer technician works on a computer, a mouse, keyboard, and graphics cable must be connected before a computer is ready to operate. Still, the Raspberry Pi device would be able to start working immediately as soon as this device is plugged into a computer. Computer technicians could plug this device into a computer they will work on while setting up their workspace. Additionally, this device would work on computers after working hours, allowing the customer to pick up their repaired PC in the morning without having to re-check for any additional updates.

Research Questions

Q1: How much can the automated Raspberry Pi Zero 2 W impact workplace performance?

Q1.1: How many computers can the automated Raspberry Pi Zero 2 W conduct maintenance on per workday day?

Q1.2: How much time can technicians save by conducting computer maintenance with the automated Raspberry Pi Zero 2 W?

Q2: How accurate is an automated Raspberry Pi Zero 2 W at automating maintenance tasks?

Hypotheses

H1: Using an automated Raspberry Pi Zero 2 W in computer maintenance will increase workplace performance by at least 10%.

H1.1: The automated Raspberry Pi Zero 2 W will be able to conduct maintenance on at least two computers per workday.

H1.2: The automated Raspberry Pi Zero 2 W will reduce the time technicians conduct computer maintenance by 48 minutes (10% of an eight-hour workday).

H2: The automated Raspberry Pi Zero 2 W will be able to perform computer maintenance tasks with an accuracy of at least 99%.

Significance of the Study

This study is significant as it develops a cost-effective automated computer maintenance tool that can be executed with a Raspberry Pi Zero 2 W (RPZ2W) to enhance workplace productivity, reduce maintenance costs, and increase daily conducted computer maintenance in computer repair offices. Businesses with computer repair technicians who spend significant time conducting repetitive maintenance tasks need this tool to increase workplace productivity since this tool will allow more time to be spent on repairing computers. When this tool is implemented in computer repair offices, the automated RPZ2W will handle most, if not all, repetitive computer maintenance tasks, allowing computer technicians to productively work on more computer repair tasks that require more physical and cognitive skills (Pourhadi, 2019; Shin et al., 2024). This reallocation of time spent on computer repair and maintenance increases technicians' workplace productivity, increases the daily possible volume of conducted computer maintenance, and decreases the labor cost of computer maintenance. Additionally, computer technicians can keep up with unexpectedly busy workdays with this tool (Brouster, n.d.).

Also, this study has significant contributions to research in the computer repair and maintenance field. Firstly, this study offers a unique way to automate the redundant computer maintenance process by developing an automation script and using a Raspberry Pi Zero 2 W to execute that script on a computer. Secondly, this study offers new insight into the feasibility of using automation tools in computer maintenance to benefit computer repair businesses financially. These contributions contribute to the overall understanding of the implications and financial impact of using automated computer maintenance tools in computer repair offices.

According to Zilberman and Ice (2021), computer occupations are expected to grow by 11.5% from 2019 to 2029 due to the quickly developing field of new technology that increases the demand for skilled professionals. Therefore, this study will retain relevancy, adding to its significance.

Definition of Key Terms

Automation Script

A program that runs on a machine to collect, send, and modify data to perform tasks that would otherwise need to be done manually (Pourhadi, 2019).

Automated Raspberry Pi Zero 2 W (RPZ2W)

A Raspberry Pi Zero 2 W that has the study's automation script on it.

Raspberry Pi Zero 2 W (RPZ2W)

A lightweight, miniature Linux-based computer that runs its own operating system and has a multifunctional USB port that can execute mouse and keyboard events (Aboluhom & Kandilli, 2024; Contino, 2024).

Computer Maintenance

Computer maintenance is the tasks technicians conduct after computer repairs to ensure proper functionality. These tasks include applying updates, installing software, optimizing systems, and keeping documentation of applied maintenance (Andersen & MoldStud Research Team, 2024; Bika, 2022; Hanna et al., 2014; Pourhadi, 2019; Urbitsch, 2023; Wingate, 2016).

Computer Repair

While different from computer maintenance, computer repair is the actions computer technicians perform to fix a computer issue rather than prevent future computer issues (Bika, 2022).

Overfitting

Overfitting occurs when a model is trained on sample data that is not representative of the general population, limiting the model's ability to successfully complete each task (Al-Bataineh et al., 2021).

Repetitive Computer Maintenance Tasks

Repetitive computer maintenance tasks in this paper are the daily computer maintenance tasks that technicians can automate with automation scripts (Pourhadi, 2019).

Summary

A Windows computer update can take up to six hours to install which encourages individuals to delay installing updates in order to prevent their own workflow interruptions (Mearian, 2022; Rajivan et al., 2020). Uninstalled computer updates are one of the main repetitive tasks that computer technicians carry out manually when conducting computer maintenance (Urbitsch, 2023). Computer technicians could be using their skill set to repair computers that require physical and cognitive skills; instead, their time is allocated to conducting maintenance tasks that can take hours (Mearian, 2022; Pourhadi, 2019; Shin et al., 2024). The goal is to reduce the time computer technicians must spend on repetitive maintenance tasks by 10% by automating these tasks, such as checking for updates, collecting device hardware and software information, and installing necessary software (such as antivirus if not present). The reduction in time spent conducting repetitive maintenance tasks will allow computer repair offices to become more productive since technicians would be able to spend more time using their skills by repairing computers that take more cognitive and physical skill to perform (Pourhadi, 2019; Shin et al., 2024). Increased workplace productivity would save significant time, money, and resources (such as desk space) for computer repair offices. The next chapter of this study describes the current computer maintenance process, identifies inefficiencies in this process, and evaluates possible solutions to address computer maintenance inefficiencies.

Chapter 2: Literature Review

Information technology technicians handle complex and tedious daily tasks to keep businesses running smoothly each day (Bika, 2022). Particular maintenance tasks, such as installing Windows updates, can become more time-consuming than others and disrupt a consistent, streamlined workflow (Mearian, 2022; Pourhadi, 2019). Manual tasks that collect documentation and generate patch reports can be automated. Still, there is a lack of tools and time to create personalized tools to address these tedious tasks. This literature review discusses the maintenance techniques that the RPZ2W used, as well as the computer maintenance techniques currently used by computer technicians.

Creating Automation Scripts

Automation scripts can be used to automate common computer maintenance tasks (Pourhadi, 2019; Urbitsch, 2023). Common computer maintenance tasks include generating patch reports, server documentation, updating drivers, clearing temporary files, and checking for Windows updates (Pourhadi, 2019; Urbitsch, 2023). Pourhadi (2019) admits that computer technicians often lack the necessary tools, overview, and time to implement scripts to automate tasks such as patch reports and server documentation. Therefore, Urbitsch (2023) has created a Windows maintenance script that automates common maintenance tasks that clear temporary files, checks for Windows Updates, and updates outdated drivers on all Windows 10 and Windows 11 computers. Computer technicians can use Urbitsch's automation script to complete maintenance tasks without spending excessive time maintaining the script (Pourhadi, 2019; Urbitsch, 2023). Because automation scripts do not require human intervention, the computer under maintenance can be placed anywhere, and valuable desk space can be saved and allocated for more computer repairs.

Hanna et al. (2014) mention Microsoft Visual Studio as an example of a scripting tool. This scripting tool provides various debugging features, including recording and playback, which are essential for developing automation scripts. When scripts are created, they often only work under the specific environments they were tested in. As Windows updates its software, the operating system's interface may change, making automation scripts unusable (Hanna et al., 2014; Lenovo, n.d.; Mearian, 2022). Hanna et al. (2014) and Pourhadi (2019) recognize that frequently updating automation scripts can be challenging due to constant system changes. However, various scripting techniques can be utilized to reduce the frequency of necessary script maintenance (Hanna et al., 2014). One scripting technique that Hanna et al. (2014) propose is a data-driven scripting technique. A data-driven scripting technique utilizes input information from the computer technician (such as username, password, and the specific maintenance tasks to be performed) to execute a personalized script. Other scripting techniques, such as keyword-driven and linear scripting techniques, can be costly or impractical. Firstly, it is expensive to create keyword-driven scripts, as technicians must specify numerous keywords to signify which script to run. Secondly, linear scripts cannot continue executing when they encounter errors because, unlike data-driven scripts, they lack conditionals, such as if statements, to manage or correct these errors (Hanna et al., 2014). Tools such as automated program repair can identify errors during software testing by finding bug-detection patterns (Al-Bataineh et al., 2021). Data-driven automation scripts use error detection patterns during computer maintenance to systematically determine which additional steps must be taken to complete the remaining maintenance tasks (Al-Bataineh et al., 2021; Hanna et al., 2014). At the end of the automated script’s execution, time of completion, failed tasks, completed tasks, and identifying system information should be reported and documented (Andersen & MoldStud Research Team, 2024; Hanna et al., 2014; Wingate, 2016).

Using a Raspberry Pi Zero 2 W for Computer Maintenance

To initialize automation scripts on Windows computers, computer technicians must utilize a mouse and keyboard to enter credentials, install the automation script, and run the automation script. This process can be made more efficient using the Raspberry Pi Zero 2 W's (RPZ2W) functionalities (Contino, 2024). The RPZ2W is a lightweight computer with USB-gadget capabilities that emulate mouse and keyboard movements from the RPZ2W's USB port. Instead of installing an automation script on the Windows computer, an automation script can be installed on the RPZ2W. This automated RPZ2W can be connected to any Windows computer and execute maintenance tasks by sending mouse movements and keyboard strokes (Contino, 2024; thewh1teagle, 2024). Unlike automation scripts inside of the Windows computer, the automated RPZ2W can execute mouse and keyboard events when plugged into the Windows computer regardless of whether the Windows computer is signed in or not (Contino, 2024). The automated RPZ2W can function before the Windows computer logs in, as it receives power and communication from the computer. thewh1teagle (2024) provides a Python module that builds the foundation for moving the RPZ2W emulated mouse and keyboard along with example scripts. This Python module can be used to develop an automation script for the RPZ2W to automate computer maintenance tasks.

Error Prevention

A significant part of computer maintenance is preventing future errors. Many computer security errors arise due to incorrect security configurations and poor computer repair and maintenance management (Coffey, 2017). To prevent these errors, computer technicians will install and run security software and Windows updates to remove and prevent malware attacks (Bika, 2022; Lenovo, n.d.). Tools such as automated program repair can identify errors during software testing by finding error detection patterns (Al-Bataineh et al., 2021). Automation scripts can use patterns of bug detection during computer maintenance to systematically determine which additional steps need to be taken to complete the remaining maintenance tasks (Al-Bataineh et al., 2021). Inside the software code for computer programs, static detection of runtime errors determines what errors can occur before the program is executed (Ge et al., 2017). Using the same logic, an automated computer maintenance script can identify the computer's operating system type and which maintenance tasks must be performed before going through each possible maintenance task. For example, the detection software can check for legal array bounds, division by zero invalid operations, overflow and underflow errors, and more (Ge et al., 2017). The computer maintenance automation script for my paper can utilize similar techniques by verifying whether antivirus software is present before attempting to install more antivirus (Bika, 2022; Ge et al., 2017). Additionally, the script can check how recently the hard drive disk (HDD) was last defragmented before performing defragmentation (Huculak, 2022; Ge et al., 2017).

Predictive maintenance can help identify future errors based on the data available from system sensors (Ucar et al., 2024; Yazdi, 2024). Predictive maintenance can be used in computer maintenance by using collected sensor data (such as CPU temperature and number of Windows errors in the last month) to predict possible future errors. Also, Windows updates use predictive maintenance measures (Lenovo, n.d.; Ucar et al., 2024; Yazdi, 2024). Lenovo (n.d.) clarifies that when Windows computer technicians identify any future or current system errors, Microsoft issues a quality update to Windows computers to address these errors. Windows quality updates offer security patches and bug fixes to prevent further mistakes and damage to existing and future Windows computers (Lenovo, n.d.).

Mitigating Overfitting

According to Al-Bataineh et al. (2021), the current issue in automated program repair scripts is overfitting. Overfitting happens in current automated program repair tools when the tool is trained on sample data that is not representative of the general population, which limits its ability to fix various issues. To minimize overfitting in my automated computer maintenance script, I must use a sample of Windows computers that are representative of the population to test and develop my script (Al-Bataineh et al., 2021). Because each task may not execute identically on each Windows computer in the population, Hanna et al. (2014) employ control and calling structures in their structured scripts to verify whether each task was completed successfully. If a task fails to execute successfully, the script can execute additional actions in response to correct the failed task (Hanna et al., 2014).

Issues with Current Computer Repair and Maintenance Models

Traditionally, computer maintenance tasks include analyzing computer performance, repairing hardware and software issues, and applying upgrades, bug fixes, and patches to systems (Wingate, 2016). In automated repair programs, Al-Bataineh et al. (2021) argue that the same number of instructions needs to be executed in direct integration, which can induce unnecessary wear and tear on the system. Instead, Al-Bataineh et al. (2021) propose that indirect integration can automatically extrapolate patterns of previously detected bugs, as it is sometimes done manually. Some current computer repair models identify bugs by going through the most commonly occurring bugs manually, which can prove to be a tedious and time-consuming task that can be done automatically with the APR tools provided above (Al-Bataineh et al., 2021). Not only can these manual tasks be time-consuming, but they can also be error-prone: Coffey

(2017) points out that implementing technological safeguards on computers can help prevent them from becoming error-prone in the future. An example of a safeguard that improves the security of a personal computer is by turning on the automatic standby lock which can be done automatically with onboard software (Coffey, 2017).

In computer maintenance models, reactive maintenance models are commonly used among technological and manufacturing industries (Yazdi, 2024). According to Yazdi (2024), instead of relying solely on reactive maintenance, emerging technology, such as artificial intelligence, can be used in proactive and predictive maintenance alongside reactive maintenance to reduce downtimes and workflow interruptions. Ucar et al. (2024) support emerging technology as traditional predictive maintenance models that lack the use of AI do not have the same high level of accuracy and performance that an AI based model has due to the lack of stability and scalability these traditional predictive maintenance models have. Although traditional models have less computational cost, AI based predictive maintenance models can more effectively scale with bigger and more complex data. AI is also able to adapt better to more unfamiliar environments/data and can be trained to factor in for more variables (Ucar et al., 2024). Since computer repair technicians can be repair multiple computers at a time, instead of installing programs on every single computer, this lightweight device (Raspberry Pi) can be used to execute complex tasks that call for AI in an efficient manner using MobileNetV2 architecture (Aboluhom & Kandilli, 2024).

Computer Maintenance Techniques

Predictive maintenance utilizes sensor data and inputs that information into intelligent systems such as artificial intelligence (AI). This AI will when analyze the sensor data to identify potential anomalies and schedule maintenance to prevent these anomalies from turning into system failures (Lima et al., 2021). For preventive maintenance, time-based and usage-based preventive maintenance are the two most used types of preventive maintenance (Yazdi, 2024). Time-based PM is executed on time intervals (weekly, monthly, annually, etc.) regardless of machine usage. Usage-based PM is executed based on the usage frequency of the system or when the system has completed a specific number of operations. Usage-based PM is often more cost-effective than time-based PM (Yazdi, 2024). For Microsoft computers, Windows updates add more functionality features and offer quality updates that that improve the performance, security, and optimization to the computers these updates are installed on (Mearian, 2022). Furthermore, Microsoft's "Patch Tuesday" is a scheduled update that gets released every second Tuesday of each month which acts as a type of schedules maintenance (Lenovo, n.d.).

Cost Savings of Computer Maintenance

Lima et al. (2021) finds that the use of predictive maintenance will lead to significant financial savings for businesses since predictive maintenance will reduce unexpected downtime from system failures and schedule tailored maintenance based on the specific errors predicted. For preventive maintenance, Yazdi (2024) claims that significant money is saved in systems that employ preventive maintenance techniques because systems that use preventive maintenance will last longer which delays the expensive replacement cost in exchange for a cheap maintenance check. Preventive maintenance can also reduce the amount of downtime when unexpected failures occur since PM identifies and replaces faulty parts before these parts fail (Yazdi, 2024; Afefy 2012; Basri et al., 2017). Preventive maintenance will also lead towards improved system reliability, system availability, and maintenance resources (Afefy, 2012; Basri et al., 2017). Applying maintenance on personal computers by performing updates can prevent costly damages from malware attacks (Rajivan et al., 2020).

Computer Maintenance Tasks Performed by Computer Technicians

According to Bika (2022), maintenance tasks that computer technicians are responsible for are setting up hardware, installing and configuring software and drivers, and managing network components such as routers and WAN/LAN. Computer technicians also perform security maintenance tasks such as managing security software in computers and networks to ensure up-to-date cyber-protection, applying regular system upgrades and updates, identifying and providing solutions to system failures, scheduling maintenance to improve system efficiency, and documenting detailed repair and maintenance reports (Bika, 2022). Improper maintenance performed by computer technicians due to human oversight/error can lead to security vulnerabilities in the future (Al-Bataineh et al., 2021). Microsoft computer technicians automatically roll out updates for Windows computers that offer improved security, functionality, and performance to maintain their client's computers (Lenovo, n.d.). The elements of predictive maintenance that are performed by maintenance technicians are testing, inspection, servicing, adjustment, installation, and calibration (Afefy, 2012). Pourhadi (2019) also agrees that technological tasks that keep businesses running smoothly should be done by IT technicians.

Artificial Intelligence in Computer Repair Tools

Ucar et al. (2024) implement AI in predictive maintenance models that can help identify future computer errors based on the data the model collects from sensors. Although traditional predictive maintenance systems can be more computationally cost effective, AI systems can analyze a wider range of data that is bigger and more complex. Alongside using AI with automated program repair tools, with static bug detection prior to automated program repair, analysis of where the bug in a program is happening can help set constraints for what the automated program has to repair by taking slices of the program where the bug is occurring which greatly reduces the time spent repairing that issue since AI can interpret a list of requirements and impose constraints on what the program needs to access (Al-Bataineh et al., 2021; Ucar et al., 2024). You can combine and use all of this kind of AI software and execute it on a lightweight device called a Raspberry Pi (Aboluhom & Kandilli, 2024). These researchers used deep learning on a Raspberry Pi (a lightweight computer) to find how effective using deep learning on a resource-limited device like this can be. MobileNetV2 was able to perform at a higher level than the Inception V3 architecture since the MobileNetV2 had a training accuracy of 98.20% and a 98% F1 score compared to the 86.80% and 91% respective performance the Inception V3 had (Aboluhom & Kandilli, 2024).

Summary

Computer technicians perform many maintenance tasks which can take refined problem-solving skills or require persistence in manual maintenance tasks (Bika, 2022; Pourhadi, 2019). The types of maintenance conducted on systems are reactive, preventive, proactive, and predictive maintenance which work together to help prevent, fix, and predict system errors and failures (Lima et al., 2021; Ucar et al., 2024; Yazdi, 2024; Basri et al., 2017; Afefy, 2012). These maintenance systems have financial benefits as these maintenance techniques reduce maintenance costs, downtime, and failure rates (Lima et al., 2021; Yazdi, 2024; Afefy, 2012). Using these maintenance techniques, a device can be created to help computer technicians automate parts of their maintenance tasks that do not require refined cognitive skills.

Advanced automated repair programs can help technicians automatically identify and fix computer issues like a RPZ2W could identify which maintenance tasks need to be applied (Al-Bataineh et al., 2021). Many other types of technology and software can be used in symphony to detect and prevent errors currently and in the future. Tools exist that can patch and fix computers, but they are for businesses that manage employees instead of customers.

Chapter 3: Research Method

This paper explores the main issue of computer maintenance technicians spending excessive time conducting repetitive maintenance tasks that take away from the time these technicians could be performing tasks requiring refined physical and cognitive skills instead of tasks requiring refined physical and cognitive skills (Pourhadi, 2019; Shin et al., 2024). This study proposes an automation script for a device, a Raspberry Pi Zero 2 W, that can be connected to a computer and execute repetitive maintenance tasks automatically. The research questions this paper aims to answer are: How can Raspberry Pi Zero 2 W impact workplace performance, and how can a Raspberry Pi be effectively used to automate maintenance tasks by receiving and transmitting input from a computer running a Windows operating system? The research hypotheses are: Using a Raspberry Pi Zero 2 W in computer maintenance will increase workplace performance by at least 10% when considering the time saved, additional completed tasks, and financial savings (H1), and the Raspberry Pi Zero 2 W will be able to perform computer maintenance tasks and generate maintenance reports automatically using its USB gadget capabilities (H2).

This study developed a script for the Raspberry Pi Zero 2 W to automate common repetitive maintenance tasks conducted by computer technicians on Windows computers. The Raspberry Pi device is programmed to simulate mouse and keyboard strokes to a Windows computer and receive HDMI input from the computer's screen. After I developed an automated computer maintenance script for the Raspberry Pi Zero 2 W (automated RPZ2W), I tested the automated RPZ2W's effectiveness. I used the results of the tests to predict how much time automated RPZ2W can save for computer technicians and evaluate the automated RPZ2W's significance.

Research Design and Methods

I used an engineering research design and a mixed methods approach to develop and test a script for the RPZ2W to automate repetitive computer maintenance tasks. Scripts are programs that can automate repetitive computer maintenance tasks to increase workplace productivity (Pourhadi, 2019). Installing scripts can be inefficient, as computer technicians must utilize a mouse and keyboard to enter credentials, install the automation script, and run the automation script (Hanna et al., 2014; Pourhadi, 2019). This process can be made more efficient by installing the automation script on the Raspberry Pi Zero 2 W (RPZ2W) that utilizes the RPZ2W's functionalities to perform computer maintenance (Contino, 2024). Unlike automation scripts installed on the Windows computer, the automated RPZ2W can receive power and execute maintenance tasks when plugged into the Windows computer, regardless of whether the Windows computer is signed in or not (Contino, 2024). Therefore, this study develops a script for the RPZ2W to automate common computer technician maintenance tasks.

To create the script, I used Microsoft's Visual Studio Code, a coding software tool, since this software is used to develop scripts in other scripting studies to achieve similar tasks (Hanna et al., 2014). Python was the primary programming language for this script since the packages I found that utilize the multifunctional USB port on the RPZ2W to simulate mouse and keyboard events are coded in Python (thewh1teagle, 2024). The multifunctional USB port on the RPZ2W allows me to replicate computer maintenance tasks as if a technician were conducting them.

To determine the most common and important maintenance tasks to test, I used qualitative data collection of maintenance checklists from computer repair companies and credible resources (Andersen & MoldStud Research Team, 2024; Bika, 2022; Hanna et al., 2014; Pourhadi, 2019; Urbitsch, 2023; Wingate, 2016). The computer maintenance tasks that the RPZ2W will conduct are checking for Windows updates, installing software, clearing temporary files, defragmenting hard drives, and documenting results and system hardware and software information, as these tasks are performed by other computer technicians (Andersen & MoldStud Research Team, 2024; Bika, 2022; Hanna et al., 2014; Huculak, 2022; Pourhadi, 2019; Urbitsch, 2023; Wingate, 2016). Since part of the engineering and design process is testing the solution, I used quantitative data collection to measure overfitting and the effectiveness of using an RPZ2W as an automation tool (Al-Bataineh et al., 2021; Hanna et al., 2014). The quantitative variables that will be used to test the effectiveness of the automated RPZ2W for the engineering and design process are the duration of each maintenance task, the number of failed and completed maintenance tasks, and the average time technicians spend each day repairing computers and conducting maintenance each day (Andersen & MoldStud Research Team, 2024; Hanna et al., 2014; Wingate, 2016).

This study has some limitations. Firstly, this study only applies to Windows 10 and Windows 11 computers, limiting the population of computers. Secondly, this study is limited by the processing power of the automated RPZ2W, as it is a lightweight computer (Aboluhom & Kandilli, 2024; Contino, 2024). Therefore, when the automated RPZ2W analyses data from the computer's screen when completing maintenance, it takes a few seconds to extract and identify that data, limiting the automated RPZ2W's effectiveness.

There are multiple assumptions in this study. This study assumes that the common maintenance tasks that computer technicians perform are applying Windows updates, installing software, optimizing systems, and keeping documentation of applied maintenance (Andersen &

MoldStud Research Team, 2024; Bika, 2022; Hanna et al., 2014; Pourhadi, 2019; Urbitsch, 2023; Wingate, 2016). Although these are the most common maintenance tasks performed, there are likely other computer technicians who perform different or more maintenance tasks that are not automated by my study's script. Secondly, to keep a consistent metric, this study assumes that computer technicians work the average eight-hour shift since technicians have day-to-day responsibilities and can be called in when there is a significant technical problem (Bika, 2022).

Setting

This research study was conducted at True-IT Inc., a local computer repair office. This company has two computer repair technicians who work eight-hour shifts and a combination of seven Windows 11 and Windows 10 computers. True-IT Inc. technicians set up computers and perform technical repairs and maintenance for various clients. Because these technicians spend significant time performing computer maintenance on clients' computers, this sample of computer technicians' work hours is appropriate for this study, considering the problem and purpose of this study. The sample of computer technicians' work hours is from a convenience sample, which assumes that the population of computer technicians work eight hours each day and lowers the study's validity of the automated RPZ2W's impact. Although many computer technicians have day-to-day responsibilities with fluctuating work hours, I chose to generalize the time computer technicians spend working each day to represent a majority of the computer technician population (Bika, 2022).

The five-size sample of maintenance tasks performed was collected from online sources during the literature review and aligned with the maintenance tasks computer technicians performed at True-IT Inc. (Andersen & MoldStud Research Team, 2024; Hanna et al., 2014; Huculak, 2022; Urbitsch, 2023; Wingate, 2016). The data-collecting process for each maintenance task in the sample involved selecting commonly performed maintenance tasks, which introduced an assumption that computer technicians only performed a combination of those five maintenance tasks. Although computer technicians perform more than five maintenance tasks, I chose to limit the number of maintenance tasks technicians perform to create an automation script that would satisfy the needs of most computer technicians to increase my study's validity (Andersen & MoldStud Research Team, 2024; Hanna et al., 2014; Huculak, 2022; Urbitsch, 2023; Wingate, 2016). Therefore, the delimitation of this study to limit the number of maintenance tasks performed made it possible to develop a script for the RPZ2W and test its accuracy and efficiency. The number of successful and unsuccessful maintenance tasks was determined by whether the execution of the task failed and if the task performed the correct actions and provided the right results. I assumed a task failed if the automated RPZ2W executed without error but performed the incorrect maintenance actions. Therefore, a delimitation of this study was to simplify the execution of a maintenance task to a pass or failure such that a single value for the passing rate of the automation RPZ2W could be used as a metric of its accuracy. The automated RPZ2W used its own internal timer to calculate the time spent conducting each maintenance task with second precision (Contino, 2024). Therefore, it is assumed that this duration data is very accurate and reliable with high validity. Another delamination of this study limits the durations recorded to seconds instead of milliseconds due to the automated RPZ2W second precision in an experiment that can take minutes to hours. To maintain the duration of computer maintenance, another delimitation was introduced that only recorded the duration of computer maintenance after the RPZ2W since each computer has a different port layout, the time spent attaching the RP2W to each computer was variable and thus not included in the computer maintenance duration.

Materials and Instruments

This research uses a Raspberry Pi Zero 2 W (RPZ2W), a lightweight computer capable of emulating various USB devices (mouse, keyboard, storage device) and analyzing screen input from its camera port by using an HDMI (High Definition Multimedia Interface) to CSI (Camera Serial Interface) adapter by Waveshare (HDMI to CSI Adapter, n.d.; Raspberry Pi, 2024). This adapter converts the HDMI signal from the host computer into video data that is readable to the RPZ2W. To reiterate, this study develops an automation script for the RPZ2W to automate computer maintenance tasks more effectively (Contino, 2024). Unlike automation scripts installed on the Windows computer, the automated RPZ2W can receive power and execute maintenance tasks when plugged into the Windows computer, regardless of whether the Windows computer is signed in or not (Contino, 2024). For the power supply and mouse and keyboard output, the RPZ2W uses two USB type A to USB mini adapters, one to transfer power, and the other to transfer mouse and keyboard events (Contino, 2024). The HDMI input to the RPZ2W uses an HDMI to mini-HDMI adapter. To perform maintenance tasks, the device will be connected to any Windows 11 or Windows 10 desktop and laptop.

To develop the automation script for the RPZ2W, I used Microsoft's Visual Studio Code since this software is used to develop scripts in other scripting studies to achieve similar tasks (Hanna et al., 2014). With Visual Studio Code, I created the automation script using the Python coding language since the Zero-HID Python package utilized the RPZ2W's functionalities to emulate emulating mouse and keyboard events (thewh1teagle, 2024). mkfs to emulate a USB storage device, pytesseract for optical character recognition, Python's CSV library for analyzing pixel colors, and FFMPEG for capturing images from the HDMI's video stream. This project also uses Ninite's software to install multiple software automatically. I will use the current maintenance checklist at my internship to guide the actions that the RPZ2W executes. Due to size and computational limitations, this device only works on computers running Windows 10 and Windows 11, which makes this study's results more reliable for Windows computers.

Operational Definition of Variables (Quantitative/Mixed Studies Only/if necessary)

Effectiveness

Effectiveness in this study is measured by accounting for the automated RPZ2W's accuracy and efficiency. If the automated RPZ2W has an accuracy of 95% and saves technicians 10% more time by automating repetitive computer maintenance tasks, then the automated RPZ2W would be effective (Hanna et al., 2014; Pourhadi, 2019).

Data Collection, Processing, and Analysis

This study will analyze video data from the host computer through an HDMI to CSI adapter. The HDMI to CSI input will collect button location data on the host computer as well as data from the computer's System Information page to collect identifying computer information such as MAC (Media Access Control) Address, operating system type, processor information, memory size, slots of memory installed, storage size, network adapter settings (including Bluetooth), desktop name, and device manufacturer (Andersen & MoldStud Research Team, 2024; Hanna et al., 2014; Wingate, 2016). The procedure to create this RPZ2W device would be first to use the Raspberry Pi Imager to install the configuration of the Raspberry Pi OS Lite (Bookworm) 64-bit using a 64-gigabyte microSD card. The custom configuration would involve enabling SSH (Secure Shell) and setting a username and password for the RPZ2W. The following lines are commands that are entered into the RPZ2W after its configuration:

```
sudo apt-get install -y git python3-pip python3-venv ffmpeg tesseract-ocr libtesseract-dev
```

```
git clone https://github.com/GameWire9/zero-hid

cd zero-hid/usb_gadget

sudo ./installer

y


v4l2-ctl --set-edid=file=zero-hid/1080p30edid

python3 -m venv ~/venv

v4l2-ctl --set-dv-bt-timings query

source ~/venv/bin/activate

pip3 install zero-hid opencv-python pytesseract

dd if=/dev/zero of=~/usbdisk.img bs=64k count=8192

mkdosfs ~/usbdisk.img

sudo nano /usr/bin/init_usb_gadget


FILE=/home/iChaseBank/usbdisk.img

mkdir -p ${FILE/img/d}

mount -o loop,ro, -t vfat $FILE ${FILE/img/d}

mkdir -p functions/mass_storage.usb0

echo 1 > functions/mass_storage.usb0/stall

echo 0 > functions/mass_storage.usb0/lun.0/cdrom

echo 0 > functions/mass_storage.usb0/lun.0/ro

echo 0 > functions/mass_storage.usb0/lun.0/nofua

echo $FILE > functions/mass_storage.usb0/lun.0/file
```

```
ln -s functions/mass_storage.usb0 configs/c.1/
[Ctrl+O Enter Ctrl+X]
sudo reboot
```

To program the RPZ2W, I used Microsoft Visual Studio Code and saved the project file to a GitHub repository. I used the Python programming language inside of Visual Studio Code to create the basis of my project since the packages required to utilize the RPZ2W's multifunctional USB to emulate mouse and keyboard outputs are coded in Python (thewh1teagle, 2024). I started off by importing Zero-HID for emulating mouse and keyboard output, pytesseract for optical character recognition, FFMPEG for capturing frames of the RPZ2W's video stream, and Python's CSV library for analyzing pixel colors. To identify if the host computer was in OS installation/setup mode, I used OCR (optical character recognition) to get the text displayed by the Windows search bar. Then, the RPZ2W will check for Windows updates and use the Ninite package to install multiple software autonomously using the RPZ2W USB emulation driver. Next, the RPZ2W will optimize the computer by clearing temporary files and defragmenting the hard drive (Huculak, 2022; Urbitsch, 2023). Then, the automated RPZ2W will open System Information and use OCR to collect identifying computer information such as MAC (Media Access Control) Address, operating system type, processor information, memory size, slots of memory installed, storage size, network adapter settings (including Bluetooth), desktop name, and device manufacturer to create a detailed patch report. Finally, after the computer reboots, the RPZ2W checks for additional Windows Updates before printing out a unique patch report for that computer. The patch report will also include the duration of maintenance work and the number of maintenance tasks that were successfully completed. Detailed information about the specific lines of code used in the automation script can be found on my GitHub repository and Appendix A: Figure 1 (Twyman, 2024).

Ethical Assurances

Although this study may seem like it has the potential to replace jobs, this device is only a tool used by computer technicians to automate the longest manual tasks that do not require much cognitive skill. Therefore, this device does not breach any ethical considerations.

Summary

This paper explores the main issue of computer maintenance technicians spending too much time conducting repetitive maintenance tasks that take away from the time these technicians could be performing tasks requiring refined physical and cognitive skills; continuation of these manual tasks leads to decreased workplace productivity and reduced time per shift (Pourhadi, 2019; Shin et al., 2024). This study used the Raspberry Pi Zero 2 W model to review the programming and implementation process of automating manual maintenance tasks on Windows computers. The Raspberry Pi device will be able to simulate mouse and keyboard strokes from the USB-A and receive HDMI input from the Windows computer screen (Raspberry Pi, 2024; HDMI to CSI Adapter, n.d.). The setting in which this engineering research design will take place is a local computer repair office. This setting was chosen due to the lack of inexpensive automation technology to patch and fix client computers non-invasively. To program the RPZ2W, I used Microsoft's Visual Studio Code and installed Python packages: Zero-HID for emulating mouse and keyboard output, FFMEG to capture images from the RPZ2W's video stream, pytesseract for optical character recognition, and Python's CSV library for analyzing pixel colors. This project also uses Ninite's software to install multiple software without going through each setup process. At the end of the computer setup, the RPZ2W will generate a patch report. The patch report includes the duration of maintenance work and the number of maintenance tasks that were successfully completed. It is assumed that only the Windows 11 operating system installation is set up. There are no ethical assurances that this research must address. Next, this paper will discuss and interpret the findings and results of the RPZ2W.

Summarize key points presented in Chapter 3 and provide supporting citations. Flow into chapter four.

Chapter 4: Findings and Results

While computer technicians spend most of their time repairing computers, they also spend significant time conducting repetitive maintenance tasks (Pourhadi, 2019). These repetitive tasks reduce the time allotted to other workplace tasks that require more cognitive and physical skills, such as repairing computer hardware, leading to decreased workplace performance (Pourhadi, 2019; Shin et al., 2024). This research utilizes the Raspberry Pi Zero 2 W (RPZ2W) functionalities to automate repetitive maintenance tasks aimed at improving workplace performance. The RPZ2W is a miniature computer with its own operating system that can emulate various USB devices when connected to another computer via USB (Contino, 2024).

This study develops a Python program to automate maintenance tasks on the RPZ2W. The program utilizes Python packages that run on the RPZ2W's operating system. The program can send mouse movements and keyboard strokes and obtain video input from the host computer using the RPZ2W's USB emulation and HDMI to CSI adapter. The maintenance tasks the RPZ2W completes are checking for updates, reinstalling operating systems (OS), installing software, clearing temporary files, and collecting system hardware and software information.

Results

The two major research questions that this study addresses are: how much can a Raspberry Pi Zero 2 W impact workplace performance (Q1), and how accurate is a Raspberry Pi Zero 2 W at automating maintenance tasks (Q2)? The first question consisted of two sub-questions: How many computers can the Raspberry Pi Zero 2 W conduct maintenance on per hour (Q1.1), and what are the long-term time and financial savings of implementing the Raspberry Pi Zero 2 W for computer maintenance offices (Q1.2)?

These two research questions determine the effectiveness and significance of implementing an RPZ2W to conduct maintenance tasks in the workplace. Q1.1 was addressed by collecting the average time the RPZ2W spent conducting maintenance tasks on computers on a convenience sample of 12 computers at a computer repair office. The following table shows the time spent on each computer.

Table 1

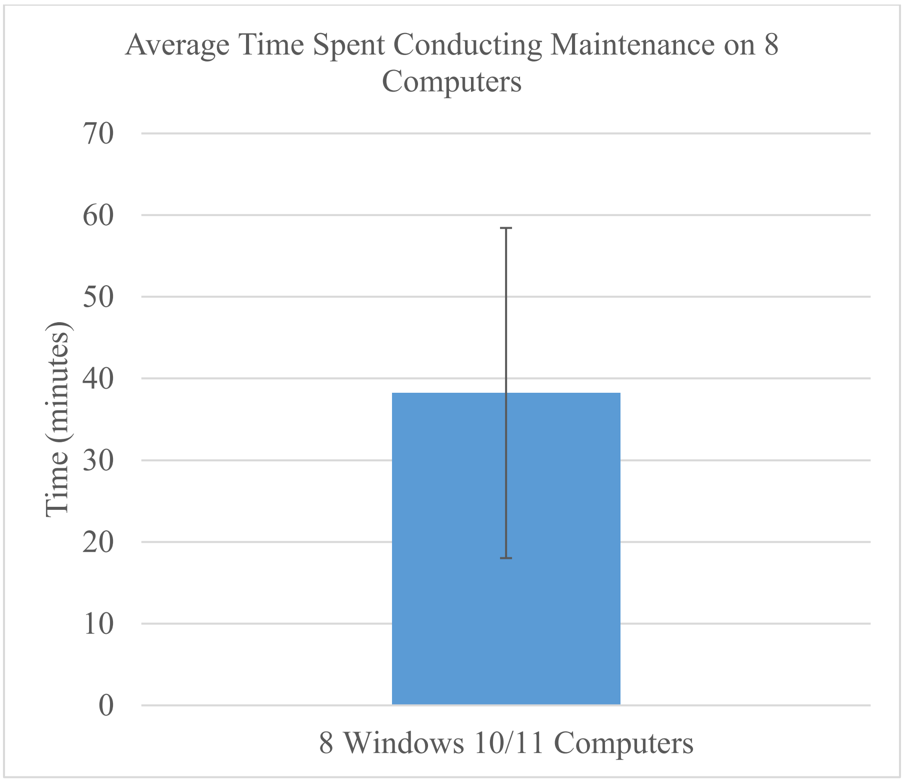
Time Spent Conducting Maintenance on 8 Computers

Computer (Brand)	Time (minutes:seconds)
Computer 1 (HP Laptop)	20:13
Computer 2 (Dell PC)	25:01
Computer 3 (ASUS Zenbook)	10:05
Computer 4 (Dell Laptop with HDD)	47:51
Computer 5 (Dell PC with HDD)	68:19
Computer 6 (CyberPower PC)	38:22
Computer 7 (HP Laptop with HDD)	67:49
Computer 8 (Lenovo Laptop)	28:08
Average Time	38:13.5
Standard Deviation	20.21

Note. Each computer was subjected to the same type and amount of maintenance tasks.

Figure 1

Bar Graph of Average Time Spent Conducting Computer Maintenance



Note. Each computer was subjected to the same type and amount of maintenance tasks.

The average time to conduct computer maintenance was used to approximate the number of computers in an eight-hour workday, a 21-workday month, and a 260-workday year. Because this device can continue to operate outside of working hours, I took the ceiling of the average instead of the floor. Therefore, computer technicians can conduct maintenance on approximately 13 computers per eight-hour workday, 273 per month, and 3380 per year. Although each computer was chosen from a convenience sample rather than a random sample, each computer was unique, with different hardware and processing power. Therefore, taking a convenience sample of computers did not significantly affect the validity of my data. The time spent on conducting maintenance for each computer was collected by the automated RPZ2W itself and started from when the automated RPZ2W ran its first maintenance task rather than when it was powered on. Since each computer has a different port layout, the time spent attaching the RP2W to each computer was variable and thus not included in the computer maintenance duration.

Table 2

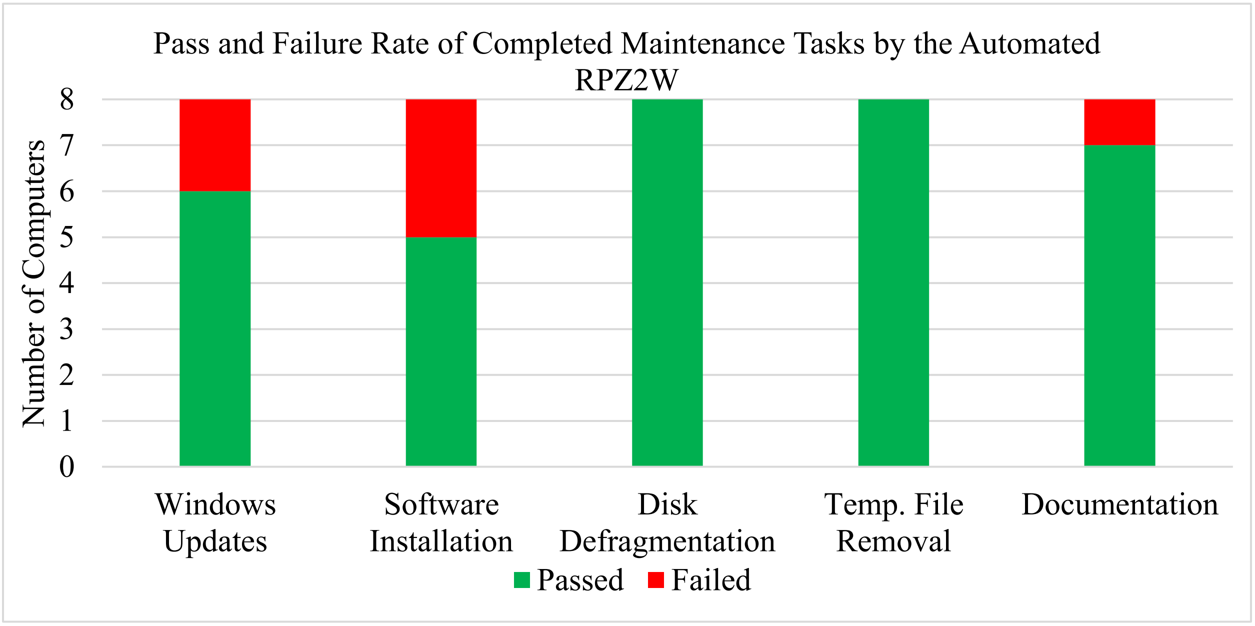
Pass and Failure Rate of Conducted Maintenance Tasks by the Automated RPZ2W

	Tasks		
Maintenance Task	Passed	Failed	Task Pass%
Windows Updates	6	2	75%
Software Installation	5	3	62.5%
Hard Drive Defragmentation	8	0	100%
Temporary File Removal	8	0	100%
Hardware and Software Documentation	7	1	87.5%
Mean			85%
Standard Deviation			16.30

Note. Passed tasks are completed tasks with the expected result, and failed tasks are incomplete or provide unexpected results.

Figure 2

Pass and Failure Rate of Conducted Maintenance Tasks by the Automated RPZ2W



Note. Passed tasks are completed tasks with the expected result, and failed tasks are incomplete or provide unexpected results. Documentation collects software and hardware information from the computer.

Software installation had the lowest passing rate while performing disk defragmentation and removing temporary files had the highest passing rate. When the automated RPZ2W fails a maintenance task, that failure is included in the maintenance report that it generates. As this device is the only kind of external Windows computer maintenance tool, there is little to compare its performance to other than its pass rate. Generally, a pass rate of at least 99% is considered acceptable since one of the aspects of automated tools is the lack of human intervention (Hanna et al., 2014). This statistical test assumes that a completed task is a task that executes completely with total accuracy, and a failed task is a task that is not completed or is completed incorrectly. The automated RPZ2W had an accuracy of 85% for passed maintenance tasks.

Evaluation of Findings

The duration of the computer maintenance was the other data recorded by the RPZ2W had millisecond precision. Instead of manually documenting how long each maintenance session took, the RPZ2W starts a timer from the time it is first initiated to when the last maintenance task is completed. Using the RPZ2W onboard computer chip to measure the duration meant there was no human error in the data collected. Since computer maintenance, on average, took 38 minutes, the quality of the time data was extremely accurate and reliable. The results obtained were expected based on the previous literature about data-driven scripting for automating computer maintenance (Hanna et al., 2014).

The data collected by the automated RPZ2W for its maintenance reports was very reliable. The RPZ2W uses optical character recognition (OCR) to translate images into text data to obtain data from a host computer. During the development and testing of the modified RPZ2W, I had no issues or errors with the quality of data the OCR collected. The OCR was 100% accurate during this research. I used the data collected by the OCR to generate documentation. When an external window obscured the system information window about the computer hardware and software information, the OCR could not read the software and hardware information and, therefore, provided an incomplete documentation report.

The number of computers that the RPZ2W could conduct maintenance on in an eight-hour workday was very reliable but not accurate. Each day, a different number of computers need computer maintenance and the RPZ2W most likely will not be in operation during those eight hours. Since the standard deviation of the duration of maintenance tasks for eight computers was 20.21, the data collected about the average time it took to conduct computer maintenance was accurate but unreliable since each computer may not need to go through each maintenance task. For instance, if a computer has the latest Windows updates, certain maintenance tasks will not be performed, shortening the duration of maintenance. Recording the average time spent on computer maintenance was the strongest metric for increasing workplace performance, and the data gave good insight into how much time the RPZ2W could save per year instead of per day.

Summary

The automated RPZ2W, on average, took 38 minutes and 13 seconds conducting computer maintenance with a standard deviation of 20.21. Therefore, computer technicians can conduct maintenance on approximately 13 computers per eight-hour workday, 273 per month, and 3380 per year. The automated RPZ2W had an accuracy of 85%. Data that was very reliable and accurate was the optical character recognition used by the RPZ2W to collect text and the duration of conducted maintenance of each computer tested. Data that had reliability but low accuracy was the approximate number of computers that the RPZ2W could perform maintenance on per day. Data that had low reliability but high accuracy was the average time it took the RPZ2W to conduct computer maintenance. Data with low reliability and accuracy was the average financial savings of the RPZ2W when implemented in a computer repair company. The data collected from the statistical analysis of the RPZ2W will be used in the next section to assess the feasibility of implementing this device in real-life applications.

Chapter 5: Implications, Recommendations, and Conclusions

To reiterate, while computer technicians spend most of their time repairing computers, they also spend significant time conducting repetitive maintenance tasks (Pourhadi, 2019). These repetitive tasks reduce the time allotted to other workplace tasks that require more cognitive and physical skills, such as repairing computer hardware, leading to decreased workplace performance (Pourhadi, 2019; Shin et al., 2024). This research utilizes the Raspberry Pi Zero 2 W (RPZ2W) functionalities to automate repetitive maintenance tasks aimed at improving workplace performance. The RPZ2W is a miniature computer with its own operating system that can emulate various USB devices when connected to another computer via USB (Contino, 2024).

This study develops a Python program to automate maintenance tasks on the RPZ2W. The program utilizes Python packages that run on the RPZ2W's operating system. The program can send mouse movements and keyboard strokes and obtain video input from the host computer using the RPZ2W's USB emulation and HDMI to CSI adapter. The maintenance tasks the automated RPZ2W completes are checking for updates, installing software, disk defragmentation, clearing temporary files, and collecting system hardware and software information.

Implications

The research questions addressed in this study were how much can the automated Raspberry Pi Zero 2 W impact workplace performance (Q1), how many computers can the automated Raspberry Pi Zero 2 W conduct maintenance on per workday day (Q1.1), how much time can technicians save by conducting computer maintenance with the automated Raspberry Pi Zero 2 W (Q1.2), and accurate is an automated Raspberry Pi Zero 2 W at automating maintenance tasks (Q2)? The hypotheses are the following: Using an automated Raspberry Pi Zero 2 W in computer maintenance will increase workplace performance by at least 10% (H1), the automated Raspberry Pi Zero 2 W will be able to conduct maintenance on at least two computers per workday (H1.1), the automated Raspberry Pi Zero 2 W will reduce the time technicians conduct computer maintenance by 48 minutes (10% of an eight-hour workday)

(H1.2), and the automated Raspberry Pi Zero 2 W will be able to perform computer maintenance tasks with an accuracy of at least 99% (H2).

Q1 evaluated how much the automated RPZ2W could impact computer technicians' workplace performance. Q1 could be answered by using the answers from Q1.1 and Q1.2. To measure the impact of the RPZ2W, Q1.1 finds the number of computers that a computer technician can conduct maintenance on using the automated RPZ2W. The more tasks that can be automated, the more time these technicians can save (Pourhadi, 2019). The statistical analysis of Table 1 answered Q1.1 since Table 1 measured the duration of the automated RPZ2W conducting computer maintenance. The results from the statistical analysis of Table 1 show that the automated RPZ2W conducts computer maintenance in an average time of 38 minutes and 13.5 seconds. The population of this study applies to computer repair technicians who work eight-hour workdays. Therefore, the automated RPZ2W can conduct maintenance on 13 computers per eight-hour workday, which answers Q1.1.

To measure the impact of the RPZ2W, Q1.2 finds the amount of time technicians can save by measuring the amount of time it takes for the automated RPZ2W to conduct five maintenance tasks. The statistical analysis of Table 1 answers Q1.2 since Table 1 measured the duration of the automated RPZ2W conducting computer maintenance. Measuring the duration of conducting these five common maintenance tasks (applying Windows updates, software installation, disk defragmentation, temporary file removal, and documentation) provided insight into the impact the automated RPZ2W could have on computer repair technician’s time spent on conducting computer maintenance (Andersen & MoldStud Research Team, 2024; Bika, 2022; Hanna et al., 2014; Pourhadi, 2019; Urbitsch, 2023; Wingate, 2016). The findings of the statistical analysis of Table 1 report that it takes the automated RPZ2W an average of 38 minutes and 13.5 seconds to conduct five computer maintenance tasks on Windows 10 and 11 computers. Thus, these findings answer Q1.2, as the automated RPZ2W reduces the time computer technicians spend on maintenance tasks by 38 minutes and 13.5 seconds.

Q2 evaluated how accurate the automated RPZ2W is when implemented in computer repair workplaces. For an automation script to be effective, it should require as little human intervention as possible (Hanna et al., 2014). Therefore, the accuracy of the automated RPZ2W should be as close to 100% as possible. Table 2 evaluates the accuracy of the automated RPZ2W by evaluating the average number of tasks that failed and passed. By measuring accuracy, you can infer the level of human intervention needed when operating this device. The average accuracy of the RPZ2W is 85% and is less than 100% needed for no human intervention (Hanna et al., 2014).

There were limitations that may have affected the interpretation of the results for Q1.2. For example, there were time-based and condition-based maintenance techniques used in the data-oriented script that did not require all maintenance tasks to be run for each conducted computer maintenance (Hanna et al., 2014; Yazdi, 204). The variations of the number of Window's updates conducted induced a large standard deviation in the duration of maintenance (20.12), which would affect the amount of time computer technicians save each day. The computers used in Table 1 to address Q1.1 were collected in a convenience sample. This may have created a disproportionate sample of the population of computers subjected to maintenance at other computer repair companies.

Recommendations

Practical recommendations for this study include running the RPZ2W on personal and business computers to update and maintain Windows Computers as this device is effective at saving time on updating and conducting maintenance on Windows devices with HDDs according to Table 1. For business use, this device can be used as a tool to assist computer repair technicians conducting computer maintenance as it saves time for the computer technician and has enough accuracy to complete basic tasks according to Figure 2. For future research, more tasks can be automated such as updating individual software, implementing AI to identify and try to fix issues with machine to improve accuracy (Figure 2), and using a more complex microcontroller than the RPZ2W that makes human interaction with the device easier than using SSH terminal to execute start maintenance tasks.

Conclusions

In this chapter, we drew logical conclusions from Q1.1, Q1.2, and Q.2 to H1.1, H1.2, and H2 based on the data collected from Table 1 and Table 2. The data from Table 1 supported H1.1 and H1.2, but the data from Table 2 did not support H2. Limitations that may have affected the results of the study include time-based and condition-based maintenance techniques used in the data-oriented script that did not require all maintenance tasks, such as available Windows updates, to be run for each conducted computer maintenance and the computers tested in this study were collected from a convenience sample (Hanna et al., 2014; Yazdi, 204). Computer repair technicians can use the RPZ2W as a tool to conduct computer maintenance and any Windows computer owning user can use the RPZ2W to maintain their computer when it is not in use. The RPZ2W can be improved further by future researchers implementing the RPZ2W with AI to improve the range of maintenance tasks it can conduct or using a different microcontroller that has a better interface than using SSH. Overall, this research paper dives into finding a way to automate computer maintenance to save time computer repair technicians expend conducting repetitive computer maintenance after each computer repair. The Raspberry Pi Zero 2 W is used to automate computer maintenance saving computer technicians an average of 38 minutes and 15 seconds for a computer with an average accuracy of 85% of carrying out each task.

References

Aboluhom, A. A. A. & Kandilli, I. (2024). Face recognition using deep learning on Raspberry Pi. *The Computer Journal*, *67*(10), 3020–3030. https://doi.org/10.1093/comjnl/bxae066

Afefy, I. (2012). *Maintenance planning based on computer-aided preventive maintenance policy* (pp. 1378–1383). Proceedings of the International MultiConference of Engineers and Computer Scientists. https://www.iaeng.org/publication/IMECS2012/IMECS2012_pp1378-1383.pdf

Al-Bataineh, O. I., Grishina, A., & Moonen, L. (2021). Towards more reliable automated program repair by integrating static analysis techniques. *IEEE 21st International Conference on Software Quality, Reliability and Security (QRS),* 654-663. https://doi.org/10.1109/QRS54544.2021.00075

Andersen, G. & MoldStud Research Team (2024, February 8). The importance of documentation in computer technician services. *MoldStud*. https://moldstud.com/articles/p-the-importance-of-documentation-in-computer-technician-services

Basri, E.I., Abdul Razak, I.H., Ab-Samat, H. & Kamaruddin, S. (2017). Preventive maintenance (PM) planning: a review. *Journal of Quality in Maintenance Engineering, 23*(2), 114-143. https://doi.org/10.1108/JQME-04-2016-0014

Bika, N. (2022, April 25). *Computer technician job description*. Workable. https://resources.workable.com/computer-technician-job-description

Brouster, G. L., (n.d.). *How to open a profitable computer repair shop?* Retrieved November 26, 2024 from https://www.thebusinessplanshop.com/en/start-a-business/guides/open-computer-repair-shop?form=MG0AV3

Coffey, J. W. (2017). Ameliorating sources of human error in CyberSecurity: Technological and human-centered approaches. *Proceedings of The 8th International Multi-Conference on Complexity, Informatics and Cybernetics (IMCIC 2017).* 85-88. https://www.iiis.org/CDs2017/CD2017Spring/papers/ZA253LY.pdf

Contino, N. (2024, October 31). *Raspberry Pi documentation: Raspberry Pi hardware.* https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

Ge, N., Jenn, E., Breton, N., & Fonteneau, Y. (2017). Integrated formal verification of safety-critical software. *International Journal on Software Tools for Technology Transfer, 20*, 423-440. https://doi.org/10.1007/s10009-017-0475-0

Hanna, M., El-Haggar, N., & Sami, M. (2014). A review of scripting techniques used in automated software testing. *(IJACSA) International Journal of Advanced Computer Science and Applications, 5*(1), 194-202. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=712f5653c4f53cea5e1f62b6668452dfb86a0feb#page=209

HDMI to CSI Adapter. (n.d.). Waveshare. Retrieved October 13, 2024, from https://www.waveshare.com/wiki/HDMI_to_CSI_Adapter

Huculak, M. (2022, December 21). *20 tips and tricks to increase PC performance on Windows 10*. Windows Central. https://www.windowscentral.com/tips-tricks-increase-pc-performance-windows-10

Lima, A. L. C. D., Aranha, V. M., Carvalho, C. J. L., & Nascimento, E.G.S. (2021). Smart predictive maintenance for high-performance computing systems: a literature review. *The Journal of Supercomputing 77*, 13494–13513 (2021). https://doi.org/10.1007/s11227-021-03811-7

Mearian, L. (2022, February 7). *Research shows Windows updates can take six hours to complete*. Computerworld. https://www.computerworld.com/article/1617577/research-shows-windows-updates-can-take-six-hours-to-complete.html

Pourhadi, S. (2019, May 17). *Top five challenges for IT technicians*. VScope. https://www.vscope.net/blog/challenges-for-it-technicians/

Rajivan, P., Aharonov-Majar, E., & Gonzalez, C. (2020). Update now or later? Effects of experience, cost, and risk preference on update decisions. *Journal of Cybersecurity, 6*(1), 1-12. https://doi.org/10.1093/cybsec/tyaa002

Raspberry Pi Zero 2 W. (2024). Raspberry Pi. https://datasheets.raspberrypi.com/rpizero2/raspberry-pi-zero-2-w-product-brief.pdf

Shin, H., Rothrock, L., & Prabhu, V. (2024). Evaluating technicians' workload and performance in diagnosis for corrective maintenance. *Sensors*, *24*(6), 1943–1943. https://doi.org/10.3390/s24061943

thewh1teagle (2024). *Zero-HID: HID python library for emulating mouse and keyboard on PI*. PyPI. https://pypi.org/project/zero-hid/

Twyman, C. (2024). *Zero-HID*. GitHub Repository https://github.com/GameWire9/zero-hid

Ucar, A., Karakose, M., & Kırımça, N. (2024). Artificial intelligence for predictive maintenance applications: Key components, trustworthiness, and future trends. *Applied Sciences, 14*(2), Article 898. https://doi.org/10.3390/app14020898

Urbitsch, M. (2023). *Windows-10-11-maintenance-script.* GitHub Repository. https://github.com/Trustiatis/Windows-10-11-maintenance-script/tree/main

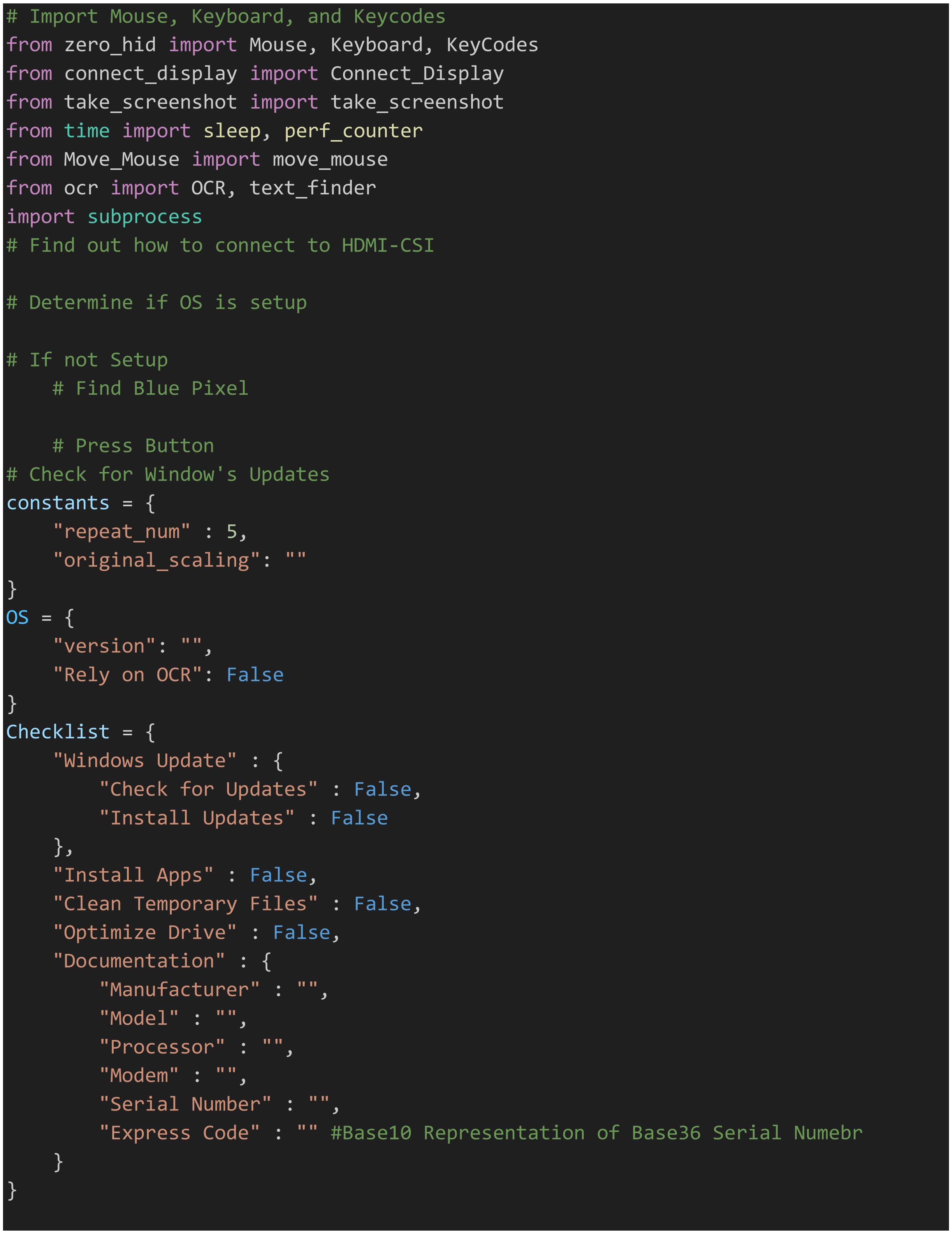
Windows updates: Everything you need to know about Windows updates. (n.d.). Lenovo. Retrieved October 13, 2024, from https://www.lenovo.com/us/en/glossary/windows-updates/

Wingate, G. (Ed.). (2016). *Pharmaceutical computer systems validation* (2nd ed., pp. 203–235). CRC Press. https://doi.org/10.3109/9781420088953

Yazdi, M. (2024). Maintenance strategies and optimization techniques. *Springer Series in Reliability Engineering*, 43-58. https://doi.org/10.1007/978-3-031-53514-7_3

Zilberman, A. & Ice, L. (2021, January). *Why computer occupations are behind strong STEM employment growth in the 2019–29 decade*. Beyond the Numbers: Employment & Unemployment, 10(1). https://www.bls.gov/opub/btn/volume-10/why-computer-occupations-are-behind-strong-stem-employment-growth.htm

Appendix A: Figure 1



```python
# Import Mouse, Keyboard, and Keycodes
from zero_hid import Mouse, Keyboard, KeyCodes
from connect_display import Connect_Display
from take_screenshot import take_screenshot
from time import sleep, perf_counter
from Move_Mouse import move_mouse
from ocr import OCR, text_finder
import subprocess
# Find out how to connect to HDMI-CSI

# Determine if OS is setup

# If not Setup
    # Find Blue Pixel

    # Press Button
# Check for Window's Updates
constants = {
    "repeat_num" : 5,
    "original_scaling": ""
}
OS = {
    "version": "",
    "Rely on OCR": False
}
Checklist = {
    "Windows Update" : {
        "Check for Updates" : False,
        "Install Updates" : False
    },
    "Install Apps" : False,
    "Clean Temporary Files" : False,
    "Optimize Drive" : False,
    "Documentation" : {
        "Manufacturer" : "",
        "Model" : "",
        "Processor" : "",
        "Modem" : "",
        "Serial Number" : "",
        "Express Code" : "" #Base10 Representation of Base36 Serial Numebr
    }
}
```

```
# Functions
def retry(function, threshold):
    for i in range(threshold):
        if function == True or function != -1:
            return true
    return false
def duplicate_display():
    k.press([KeyCodes.MOD_LEFT_GUI], KeyCodes.KEY_R)
    sleep(3) # Probably needs a conditional
    k.type("DisplaySwitch.exe /clone")
    k.press([], KeyCodes.KEY_ENTER)
    sleep(5)
    Connect_Display()
def determine_OS(not_first_time = False): # Checks for and Saves OS Version
    if (OCR(True, {"width":63, "height":23, "x":671, "y":1034},
"os_screenshot")).__enter__().lower() == "search\n": # Checks for Windows 11
Search Bar
        OS["version"] = "Windows 11"
        return True
    if OCR(True, {"width":170, "height":25, "x":107, "y":1043},
"os_screenshot").__enter__().lower() == "type here to search\n": # Checks for
Windows 10 Search Bar
        OS["version"] = "Windows 10"
        return True
    if text_finder(True, {"width":390-83, "height":1073-1027, "x":83, "y":1027},
"scaled_os_screenshot").containsNCS("Type") > -1:
        OS["version"] = "Windows 10"
        return True
    if text_finder(True, {"width":960, "height":180, "x":0, "y":900},
"larger_scaled_os_screenshot").containsNCS("Search") > -1:
        OS["version"] = "Windows 11"
        return True
    # If niether returns true, then assume computer is on login screen or not on
desktop and try to login assuming no password
    if not_first_time:
        k.press([KeyCodes.MOD_LEFT_GUI], KeyCodes.KEY_D) # Goes to Desktop if on
a fullscreen window
        sleep(1)
        k.press([], KeyCodes.KEY_SPACE)
        sleep(3)
        k.press([], KeyCodes.KEY_SPACE)
    else:
        k.press([], KeyCodes.KEY_SPACE)
        sleep(3)
        with open('passwords.txt', 'r') as file:
            k.type(file.readline())
        k.press([], KeyCodes.KEY_ENTER)
    return False
def set_scaling():
    # Open Settings
    k.press([KeyCodes.MOD_LEFT_GUI], KeyCodes.KEY_I)
    if OS["version"] == "Windows 10":
        sleep(1) # Might Need a conditional
        k.press([KeyCodes.MOD_LEFT_GUI], KeyCodes.KEY_UP)
        for i in range(10):
            if text_finder(True, {"width":175, "height":43, "x":0, "y":0}, 
"settings_corner_screenshot").containsNCS("Settings") > -1:
                break
            if i >= 9:
                return False
        system_settings_text = text_finder(True, {"width":405-141, "height":583-
315, "x":141, "y":315}, "System_screenshot")
        if system_settings_text.containsNCS("System") > -1:
            system_settings_text.goto_text(141, 315)
            for i in range(5):
                if text_finder(True, {"width":757-340, "height":165-57, "x":340, 
"y":57}, "display_corner_screenshot").containsNCS("Display") > -1:
                    break
                if i >= 4:
                    print("There was an error trying to set scaling (finding the 
display screen)")
                    return False
            for i in range(9):
                k.press([], KeyCodes.KEY_TAB)
                if i < 8: sleep(0.1)
            scaling_button = text_finder(True, {"width":681-351, "height":1013-
932, "x":351, "y":932}, "scale_button_screenshot")
            scales = ["100%", "150%", "175%"]
            if scaling_button.containsNCS("125%") == -1:
                for i in scales:
                    if scaling_button.containsNCS(i) > -1:
                        scaling_button.goto_text(351, 932)
                        scaling_button_options = text_finder(True, {"width":686-
350, "height":1005-758, "x":350, "y":758}, "scale_button_options_screenshot")
                        if scaling_button_options.containsNCS("125%") > -1:
                            location = scaling_button_options.goto_text(350, 758)
                            sleep(5)
                            k.press([KeyCodes.MOD_LEFT_ALT], KeyCodes.KEY_LEFT)
                            constants["original_scaling"] = i
                            return True
                    if i == scales[-1]:
                        print("Unsupported Display Size or Resolution")
                        return False
            else:
                constants["original_scaling"] = "125%"
                k.press([KeyCodes.MOD_LEFT_ALT], KeyCodes.KEY_LEFT)
                return True
        else:
            print("Could not find system settings button")
            return False
    elif OS["version"] == "Windows 11":
        k.press([KeyCodes.MOD_LEFT_GUI], KeyCodes.KEY_DOWN)
        sleep(0.5)
        k.press([KeyCodes.MOD_LEFT_GUI], KeyCodes.KEY_DOWN)
        sleep(0.5)
        k.press([KeyCodes.MOD_LEFT_GUI], KeyCodes.KEY_UP)
        sleep(0.5)
        k.press([KeyCodes.MOD_LEFT_GUI], KeyCodes.KEY_UP)
def check_for_updates():
    # Open Settings

    if OS["version"] == "Windows 10":
        sleep(3)
        if OCR(True, {"width":148, "height":21, "x":878, "y":658},
"update_path").__enter__().lower() == "update & security\n":
            move_mouse(74+888, 10+668)
            sleep(1)
            rel_mouse.left_click()
        else:
            whole_screen = text_finder(False, "update_path", new_image = True)
            if whole_screen.containsNCS("Security") > -1:
                whole_screen.goto_text()
            else:
                print("There was an error finding the Windows Update Settings
Icon")
                return False
        for i in range(10):
            if OCR(True, {"width":691-425, "height":115-68, "x":425, "y":68},
"windows_update_title").__enter__().lower() == "windows update\n":
                break
            if i >= 4:
                print("There was an error clicking the Windows Update Settings
Icon")
                return False
        found_text = text_finder(True, {"width":1530-400, "height":1030, "x":400,
"y":0}, "update_window")
        if found_text.containsNCS("Install") > -1: found_text.goto_text(400, 0)
        if OCR(True, {"width":1137-1059, "height":591-567, "x":1059, "y":567},
"get_latest_updates_prompt").__enter__().lower() == "not now\n":
            move_mouse(1059+(1137-1059)/2, 567+(591-567)/2)
            sleep(0.1)
            rel_mouse.left_click()
        found_text = text_finder(True, {"width":1530-400, "height":1030, "x":400,
"y":0}, "update_window")
        if found_text.containsNCS("Download") > -1: found_text.goto_text(400, 0)
        if OCR(True, {"width":592-445, "height":250-230, "x":445, "y":230},
"check_for_update_button").__enter__().lower() == "check for updates\n":
            move_mouse(445+(592-445)/2, 230+(250-230)/2)
            sleep(0.1)
            rel_mouse.left_click()
            Checklist["Windows Update"]["Check for Update"] = True
            return True
        elif found_text.containsNCS("Download") > -1:
            found_text.goto_text(400, 0)
            Checklist["Windows Update"]["Check for Update"] = True
            return True
        elif found_text.containsNCS("Downloading") > -1 or
found_text.containsNCS("Installing") > -1 or found_text.containsNCS("Required") >
-1:
            Checklist["Windows Update"]["Check for Update"] = True
            return True
        else:
            print("Could not find Check for Updates Button")
            return False

    if OS["version"] == "Windows 11":
        if not OS["Rely on OCR"]:
            with OCR(True, {"width":238-80, "height":819-794, "x":80, "y":794},
"update_path") as update_path:
                complete = False
                for i in range(0, constants["repeat_num"]):
                    if update_path.lower() == "windows update\n":
                        with move_mouse(80+(238-80)/2, 794+(819-794)/2) as
mouse_move: {}
                        sleep(1)
                        rel_mouse.left_click()
                        complete = False
                        for i in range(0, constants["repeat_num"]):
                            with OCR(True, {"width":1862-1697, "height":249-224,
"x":1697, "y":224}, "check_for_update_button") as button:
                                if button.lower() == "check for updates\n":
                                    with move_mouse(1697+(1862-1697)/2, 224+(249-
224)/2) as mouse_move: {}
                                    sleep(1)
                                    rel_mouse.left_click()
                                    complete = True
                                    Checklist["Windows Update"]["Check for
Updates"] = True
                                    break
                            if i == constants["repeat_num"] and not complete:
                                print("error clicking windows button")
                        complete = True
                        break
                    else:
                        complete = False
                    if i == constants["repeat_num"] and not complete:
                        print("error occured when checking for windows update
path")
def install_software():
    k.press([KeyCodes.MOD_LEFT_GUI], KeyCodes.KEY_R)
    sleep(3)
    k.type('"D:\\New Windows Setup.exe"')
    k.press([], KeyCodes.KEY_ENTER)
    sleep(10)
    k.press([], KeyCodes.KEY_LEFT)
    sleep(0.1)
    k.press([], KeyCodes.KEY_ENTER)
    sleep(3)
    k.press([KeyCodes.MOD_LEFT_GUI], KeyCodes.KEY_R)
    sleep(3)
    k.type('D:\\Reader_en_install.exe')
    k.press([], KeyCodes.KEY_ENTER)
    sleep(10)
    k.press([], KeyCodes.KEY_LEFT)
    sleep(0.1)
    k.press([], KeyCodes.KEY_ENTER)

with Keyboard() as k, Mouse() as rel_mouse:
    timer1 = perf_counter()
    rel_mouse.move(1,0)
    rel_mouse.move(-1,0)

    # Command 1
    Connect_Display()

    timer2 = perf_counter()
    print(f"Time to Connect Display: {timer2 - timer1:.6f} seconds\n")
    # Check if Computer is in Duplicate Display Mode

    # Command 2
    if not determine_OS() and not determine_OS(True) and not
retry(determine_OS(True), 15): print("An Error Occurred when trying to determine
OS Version")

    timer3 = perf_counter()
    print(f"Time to Determine OS: {timer3 - timer2:.6f} seconds\n")

    # Command 3
    duplicate_display()

    timer4 = perf_counter()
    print(f"Time to Duplicate Display: {timer4 - timer3:.6f} seconds\n")

    # Command 4
    if not set_scaling():
        print("An Error Occurred when trying to set scaling to 125%")
        OS["Rely on OCR"] = True

    timer5 = perf_counter()
    print(f"Time to Set Scaling: {timer5 - timer4:.6f} seconds\n")

    # Task 1
    if not check_for_updates(): print("An Error Occurred when Checking for
Windows Updates") # Check for Windows Updates
    else:
        if Checklist["Windows Update"]["Check for Update"] != True:
            Checklist["Windows Update"]["Check for Update"] = True
        k.press([KeyCodes.MOD_LEFT_ALT], KeyCodes.KEY_F4)



    timer6 = perf_counter()
    print(f"Time to Check for Windows Updates: {timer6 - timer5:.6f} seconds\n")

    # Task 2
    install_software()
    # Check later if software installed correctly
    timer7 = perf_counter()
    print(f"Time to Check for Windows Updates: {timer7 - timer6:.6f} seconds\n")
    print(f"Total Time: {timer7 - timer1:.6f} seconds\n")
```

You need to include your survey or other instrument if you have one. You need to include an appendix with a table including all of your raw data. Any figures or tables that are too large (takes up more than a half page of space) should have an appendix. All tables and figures in the body of your paper and in your appendices should be APA format including a full figure/table legend. Figures and tables are numbered within each appendix (Appendix A Figure 1 and Appendix A Table 1). If a figure or table is in the paper, it should not also be in the appendix because that is redundant.