

Hao Bai, Haob.19@illinois.edu, UIUC

VSC-WebGPU: A Selenium-based VS Code Extension For Local Edit And Cloud Compilation on WebGPU

Prerequisites

WebGPU: A Scalable Online Development Platform for GPU Programming Courses

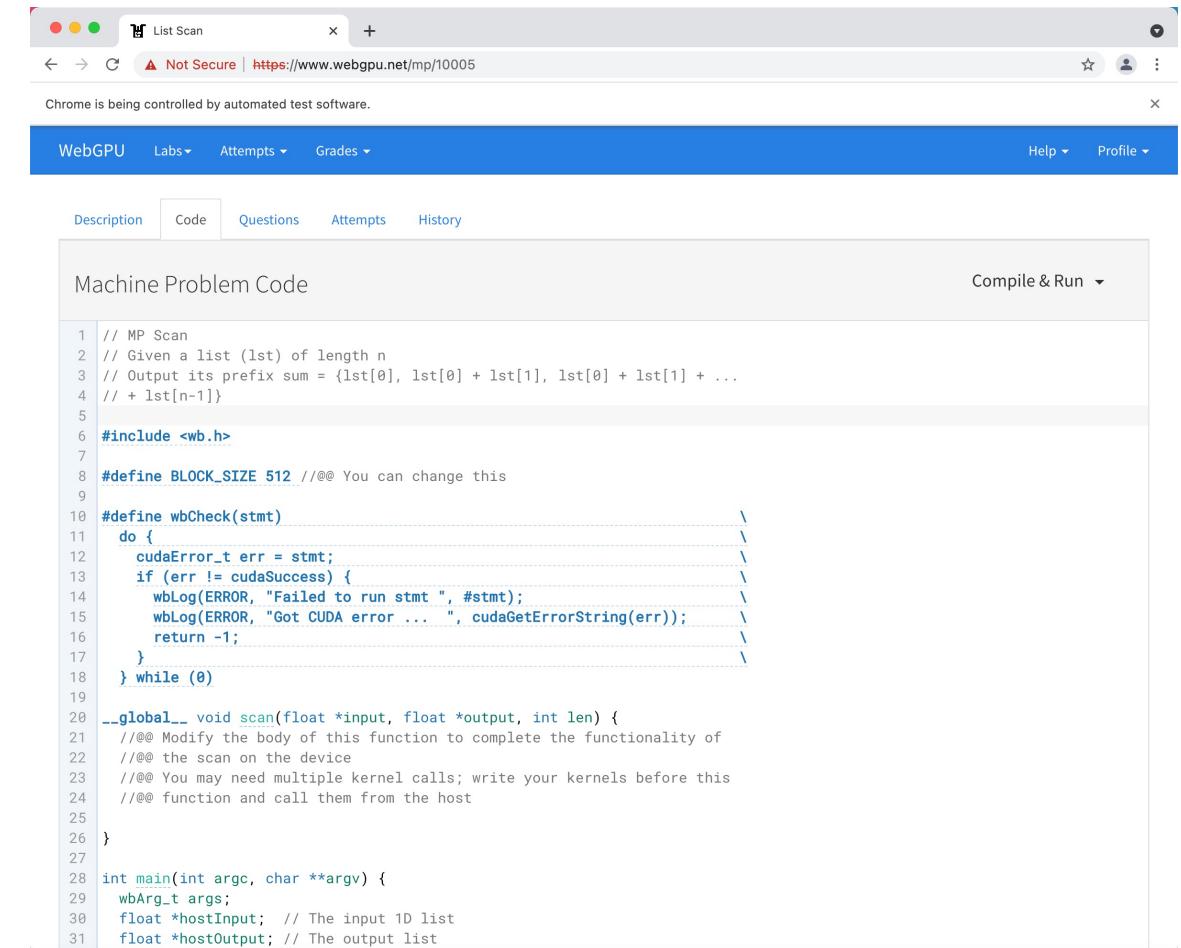
Abdul Dakkak

*Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, USA
dakkak@illinois.edu*

Carl Pearson and Wen-mei Hwu

*Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, USA
{pearson, w-hwu}@illinois.edu*

What's based on?



The screenshot shows a web browser window titled "List Scan" with the URL <https://www.webgpu.net/mp/10005>. The page is titled "Machine Problem Code". It features a code editor with syntax highlighting for C-like code. The code includes CUDA error checking and a main function that calls a global kernel. The browser status bar indicates "Chrome is being controlled by automated test software."

```

1 // MP Scan
2 // Given a list (lst) of length n
3 // Output its prefix sum = {lst[0], lst[0] + lst[1], lst[0] + lst[1] + ...
4 // + lst[n-1]}
5
6 #include <wb.h>
7
8 #define BLOCK_SIZE 512 //@@ You can change this
9
10#define wbCheck(stmt)
11    do {
12        cudaError_t err = stmt;
13        if (err != cudaSuccess) {
14            wbLog(ERROR, "Failed to run stmt ", #stmt);
15            wbLog(ERROR, "Got CUDA error ... ", cudaGetErrorString(err));
16            return -1;
17        }
18    } while (0)
19
20__global__ void scan(float *input, float *output, int len) {
21    //@@ Modify the body of this function to complete the functionality of
22    //@@ the scan on the device
23    //@@ You may need multiple kernel calls; write your kernels before this
24    //@@ function and call them from the host
25
26}
27
28int main(int argc, char **argv) {
29    wbArg_t args;
30    float *hostInput; // The input 1D list
31    float *hostOutput; // The output list

```

```
int main(int argc, char **argv) {
    wbArg_t args;
    float *hostInput; // The input 1D list
    float *hostOutput; // The output list
    float *deviceInput;
    float *deviceOutput;
    int numElements; // number of elements in the list

    args = wbArg_read(argc, argv);

    // we use the un-optimized approach
    __global__ void total(float *input, float *output, int len) {
        in
        //@@ Abbreviation Generation
        int bx = blockIdx.x;
        int tx = threadIdx.x;
```

Hao Bai(*). "VSC-WebGPU: A Selenium-based VS Code Extension For Local Edit and Cloud Compilation on WebGPU." The 2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC 2021). IEEE, 2021.

Problem

- Few syntax highlight.
- No auto-completion.

```
int main(int argc, char **argv) {
    wbArg_t args;
    float *hostInput; // The input 1D list
    float *hostOutput; // The output list
    float *deviceInput;
    float *deviceOutput;
    int numElements; // number of elements in the list

    args = wbArg_read(argc, argv);

    wbTime_start(Generic, "Importing data and creating memory on host");
    hostInput = (float *)wbImport(wbArg_getInputFile(args, 0), &numElements);
    hostOutput = (float *)malloc(numElements * sizeof(float));
    wbTime_stop(Generic, "Importing data and creating memory on host");

    wbLog(TRACE, "The number of input elements in the input is ",
          ||| numElements);
```

```
__global__ void scan(float *input, float *output, int len) {
    in
    // abc include
    // abc input
    // abc int
    // abc Initialize
```

Inspiration

VS Code can do this well!

The screenshot shows the Selenium website's homepage. At the top, there is a navigation bar with links for About, Downloads, Documentation, Projects, Support, Blog, and English. A search bar is also present. The main content area has a green background. It features a large heading "Selenium **automates** browsers. That's it!" where "automates" is highlighted with a red border. Below this, a sub-heading reads "What you do with that power is entirely up to you." A descriptive text follows: "Primarily it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should) also be automated as well."

Selenium automates browsers. That's it!

What you do with that power is entirely up to you.

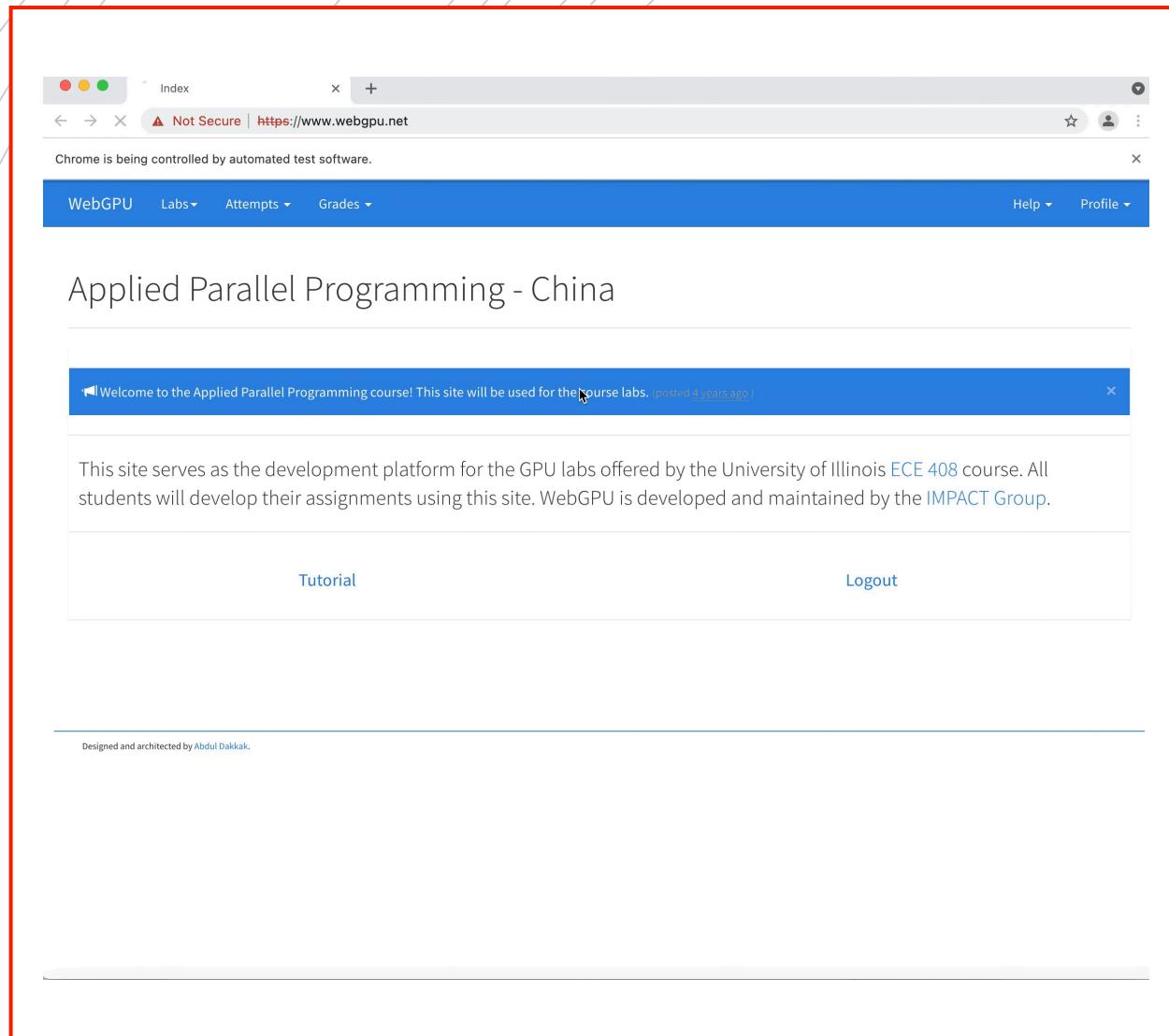
Primarily it is for automating web applications for testing purposes, but is certainly not limited to just that.
Boring web-based administration tasks can (and should) also be automated as well.

Hao Bai(*). "VSC-WebGPU: A Selenium-based VS Code Extension For Local Edit and Cloud Compilation on WebGPU." The 2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC 2021). IEEE, 2021.

The Approach: Selenium Automata

- <https://www.selenium.dev/>
- We use Selenium in VS Code extension to combine them up!

An Automated Illustration of Selenium



- Everything is done automatically.



Architecture

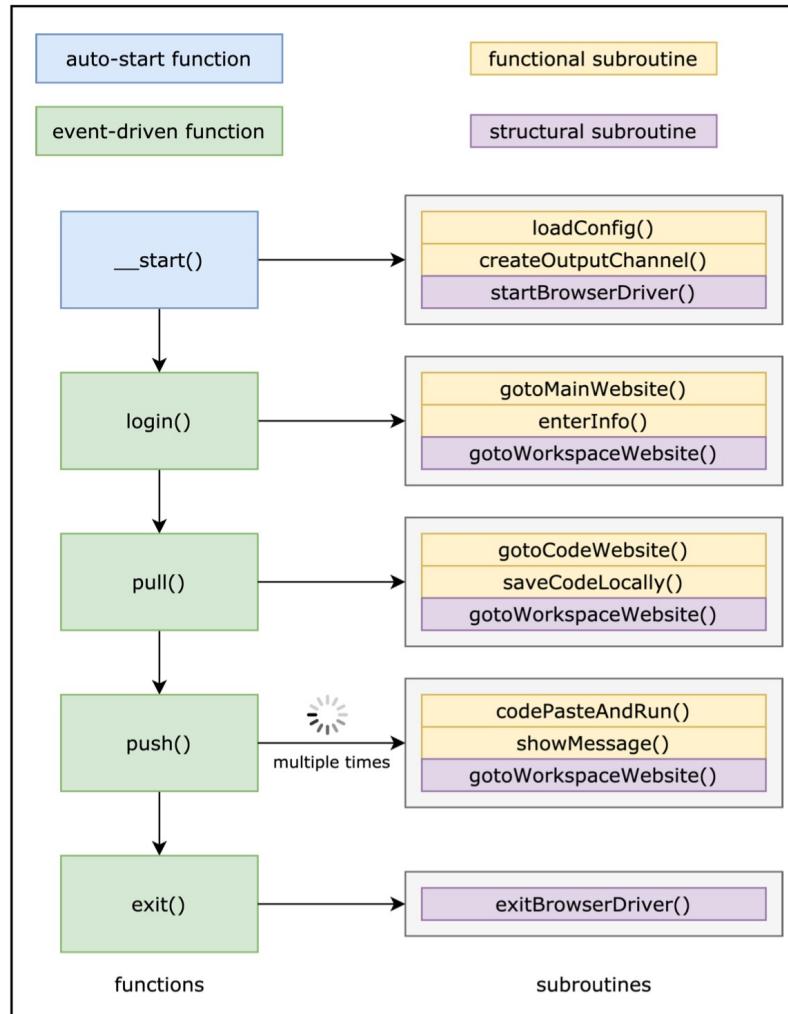


Figure 1. The overall structure of the extension. **Left:** the functions implemented in the extension. **Right:** the subroutines (functionalities) of each function. Note that the push function may be called several times, please students tend to write code and run several times.

Technical Challenges

Blocking in Node.js

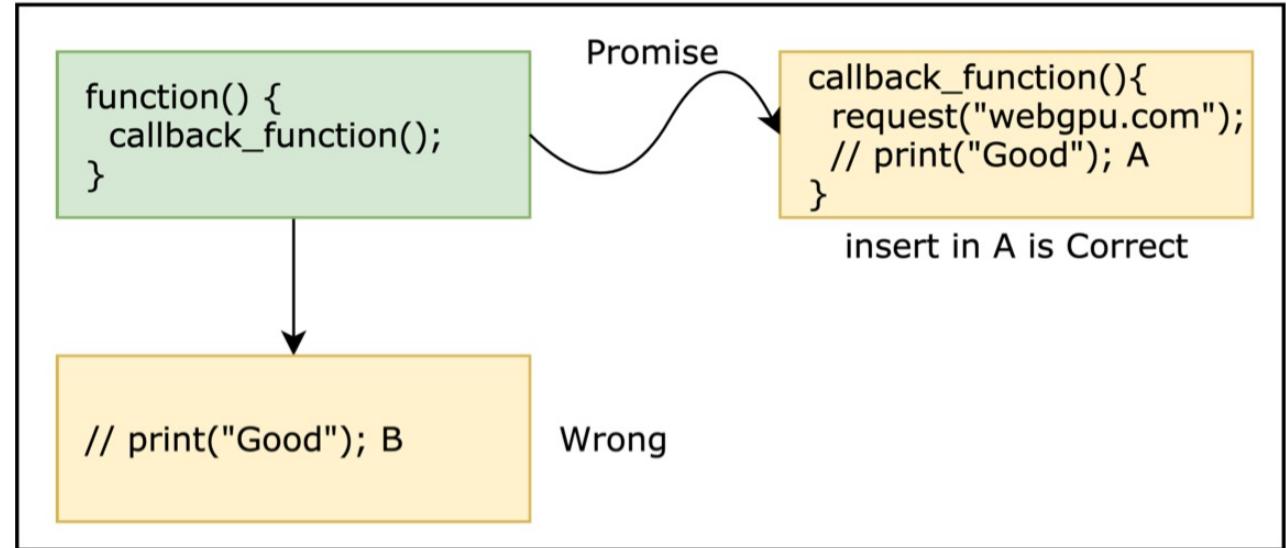


Figure 3. A comparison of blocking or not. Server programmers tend to use choice **A** to report potential errors. **A:** blocking. **B:** non-blocking.

Implicit and Explicit Waiting

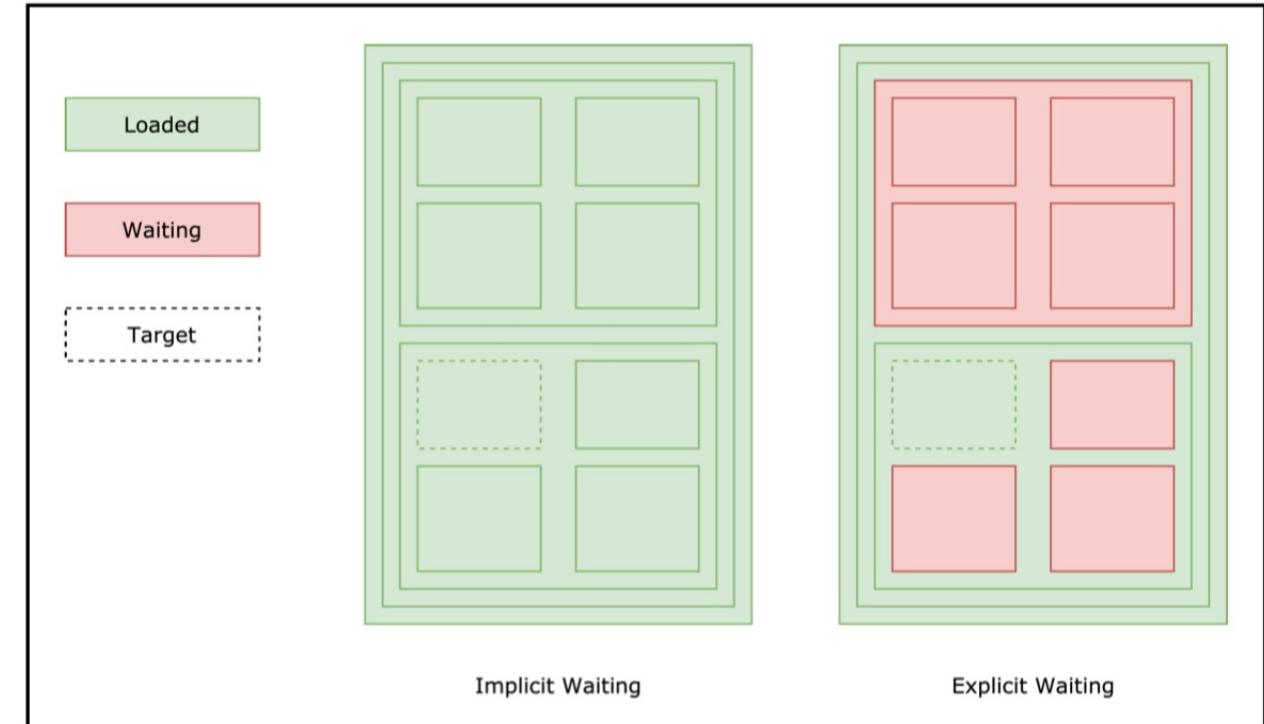


Figure 4. Difference between implicit waiting and explicit waiting. The explicit waiting method waits for only the target to be loaded to carry on the next command (non-blocking) while the implicit waiting method waits for all the elements in the web page to get loaded (blocking).



Apply to the wider world

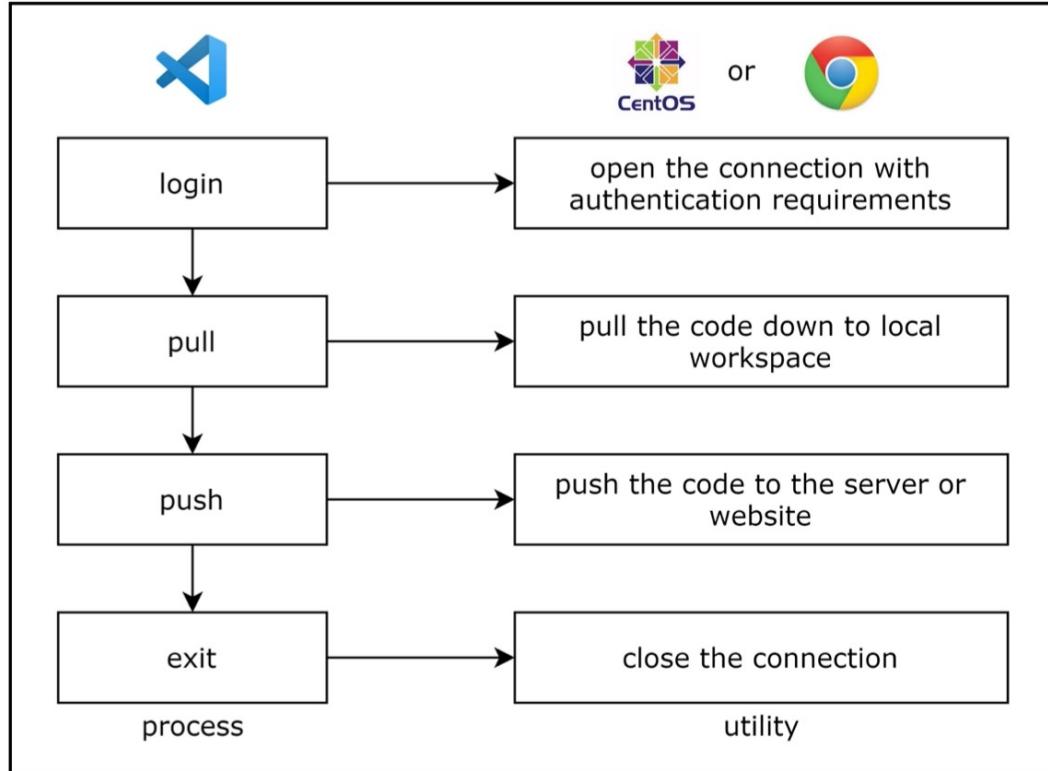


Figure 6. Our design pattern can use *VS Code* to create a connection between not only the browser but also directly the server. The workflow applies to most of the scenarios where we want to run our code on the cloud instead of a local machine.

REFERENCES

- [1] Godse, M., & Mulik, S. (2009, September). An approach for selecting software-as-a-service (SaaS) product. In 2009 IEEE International Conference on Cloud Computing (pp. 155-158). IEEE.
- [2] Cusumano, M. (2010). Cloud computing and SaaS as new computing platforms. Communications of the ACM, 53(4), 27-29.
- [3] Dakkak, A., Pearson, C., & Hwu, W. M. (2016, May). Webgpu: A scalable online development platform for gpu programming courses. In 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (pp. 942-949). IEEE.
- [4] Grigors, E. J. (2021). Beautiful Soup, Selenium and Scrapy Speed, Efficiency and Solution Comparison.
- [5] XIAO, B., & CHEN, Z. X. (2008). Headless Mode Application Study Based on JAVA. Computer Knowledge and Technology, 2008, 26.
- [6] García, B., Munoz-Organero, M., Alario-Hoyos, C., & Kloos, C. D. (2021). Automated driver management for selenium WebDriver. Empirical Software Engineering, 26(5), 1-51.
- [7] Shah, H. (2017). Node. js challenges in implementation. Global Journal of Computer Science and Technology.
- [8] Satheesh, M., D'mello, B. J., & Krol, J. (2015). Web development with MongoDB and NodeJs. Packt Publishing Ltd.
- [9] Li, R., & Hu, J. (2013). Study of Asynchronous Non-Blocking Server Based on Nodejs.
- [10] Li, R., & Hu, J. (2013). Study of Asynchronous Non-Blocking Server Based on Nodejs.
- [11] Wang, S., Keivanloo, I., & Zou, Y. (2014, November). How do developers react to RESTful api evolution?. In International Conference on Service-Oriented Computing (pp. 245-259). Springer, Berlin, Heidelberg.

Github: <https://github.com/BiEchi/ece408-remote-control>



Thanks for listening!