WIDGET LAB 1 DOCUMENT

* Show process and outcome of systems design and prototyping as thoroughly as possible
* Formatting – refer to 9.2

PART 1

* Summarize Widget Lab 1 Assignment work
  + Include diagrams/images needed to explain how system works
  + No more than 6 pages (plus 1 page reference and 2 pages appendices)
* Overview of conceptual **design process** (1-2 pages)
  + Moving from framing the system based on requirements (from section 6) to creating a conceptual design
  + Explain design process and outcome
  + Presence and clarity of system specification and/or high-level system goals
  + Artefacts
    - MUST: at least one abstract representation of system (state diagram, flowchart, behaviour table, etc. from design process)
    - Diagrams
    - Flowcharts
    - Tables
    - Pseudocode

[picture of state diagram]

To begin the design process, my lab partner and I drew a state diagram representing the functionality of the system in order to obtain a clearer picture of what the end goal was. This idea took inspiration from the state diagrams we had learned in ECE253 (Digital Logic and Computer Systems) the previous semester, as we saw the system’s parallel to finite state machines. After drawing out this parallel, we clearly saw that the system is essentially a finite state machine with three states (one for each lighting mode), where the next state of the system is determined by the value that the button changes to and the output of the system (the LEDs) is determined by the current state of the system. Although we did not explicitly draw a state table, in the process of drawing the diagram, we ended up drawing a rough state table as well. We initially labelled the system states as ‘A’, ‘B’, and ‘C’, but thinking ahead to the coding portion, we reasoned that it would make the logic for changing states much more concise if we represented the states using integers (0, 1, and 2); since the changing of states follows a very sequential pattern (0, 1, 2, 0, 1…), it would be easy to represent it using a single mathematical expression. We decided 0 represents *off*, 1 represents *plant-only*, and 2 represents *plant-and-worker*.

We used a Boolean representation for the state of the button and LEDs (using 0 or 1). Intuitively, for the LEDs, we used 1 to represent *on*, but for the button, we thought ahead to the circuit and coding portion and decided to use 0 to represent *pressed* since from Widget Lab 1 we learned that the input pins read a low state when the button is pressed. We also wrote out logic expressions for the LEDs using our state table as we anticipated that it would come in handy for the coding portion.

Code

After figuring out the functionality of the system, we immediately began writing pseudocode after since the logic of the system was fresh in our minds. We needed a variable to keep track of the state, and since the system is responding to “*a change in value of the button*” (from Widget Lab 1 Assignment document), we also needed a variable to keep track of the *previous* button value. We began an infinite loop to continuously run the code, and in the loop, we change the state and update the values of the LEDs if and only if the button has changed from *not pressed* to *pressed*. The written pseudocode is shown in Figure X.

[picture of pseudocode]

Circuit

For the circuits, our goal in the design process was to simply get an idea of what circuits needed to be built and roughly what they looked like. Since we only had two LEDs, we used the red LED to represent the red and blue light, and the green light to represent the white light. Having recently completed the Widget Lab 1, we realized that the circuitry was essentially the same as the one built in Section 12 of the lab, except there was one more LED; the only other discrepancy was the logic when the button is pressed, which is handled by the code. We drew a rough sketch for the circuitry for the LEDs and the button similar to those shown in the Widget Lab 1 document. These sketches are shown in Figure X.

[picture of circuit sketches]

Structural

After reading the system specifications for the structural component, we summarized that our task was essentially to create a case for the breadboard that covers everything except for the button and LEDs, prevents breadboard movement, and doesn’t require adhesives to assemble. We began drawing sketches of what we wanted the case to look like, shown in Figure X.

[picture of structural sketch]

Looking at the circuits built in Section 12 of Widget Lab 1 and predicting what the circuit would look like on the breadboard for the system, we reasoned that it was fully feasible to localize the visible portions of the system (LEDs and button) near the bottom of the breadboard (on the opposite end of the Raspberry Pi Pico) so that the case could have less openings and therefore be less complex (keeping in mind the fabrication process). What we had in mind resembled a box with a removable top portion, but we were unsure what fastening mechanism to use for the top portion. We brainstormed using screws and bolts, snap fit joints, etc. but ultimately, we decided it would be best to wait until we were assigned a fabrication method, as that would have a large impact on our design.

* Overview of implementation process (1-2 pages)
  + Moving from creating a conceptual design to implementing it as a physically embodied prototype
  + Explain development process and outcome
  + Justification for material/component/tool selection
  + Evidence of systematic approach to or rationale for developing code, circuit, and structural aspects
  + Evidence of consideration of integration and interfacing issues
  + Artefacts
    - MUST: at least one detailed circuit diagram from development process, at least one image of CAD model of structural element from development process
    - Circuit diagrams
    - Pictures of CAD models

Code

Implementing the code from the pseudocode was a straightforward process. Since Section 12 of Widget Lab 1 also required the configuration of the LEDs and the button, we simply copy and pasted that portion of code from Widget Lab 1. As for the logic, we went through the pseudocode line by line, translating it into Python code, while making sure we understood each line. There were some errors in our pseudocode that we fixed when writing the code, such as a missing statement to update the previous button value. While we did consider the use of functions when we wrote our pseudocode, we decided to not use any in the end due to lack of necessity. Functions help with writing cleaner and more reusable code, but in this situation, the code was already very short, clean, and readable. For example, we originally had planned to write a function that determines the next state and another to update LEDs, but these can all be written in one line in the first place. Combining all of these into one function is also unnecessary and doesn’t add to the reusability of the code.

Circuit

[picture of circuit diagram]

Before building the circuit on the breadboard, we drew a final circuit diagram, shown in Figure X. We labelled the GPIO pins and the ‘boundaries’ of the Raspberry Pi Pico so it was clear to us what we needed to build. Again, drawing parallels to Widget Lab 1, we used the 330 ohm resistors for the LEDs we used before since we knew they worked. For the wire colours, we used black for ground, red for power, and blue to connect the button to the input pin of the Raspberry Pi Pico. We first used the jumper cables when building the circuit in case it didn’t work to avoid wasting the solid core wires. Regarding the positioning on the actual breadboard, we once again took inspiration from Widget Lab 1. As mentioned in Section (section for design process of circuit), the circuits for this system should be the same as those in Widget Lab 1. Therefore, to implement the green LED, we simply mirrored the red LED circuit across the breadboard’s centre gap. Compared to making a circuit on the same side of the centre gap, mirroring is more aesthetically appealing (due to symmetry) and makes the circuits easier to follow, since the former would cause a lot of overlap between the wires, making it messy. Finally, we maximized the localization of the visual parts and moved them as far away from the rest of the circuits as possible by shifting the button all the way to the bottom and moving the LEDs to the middle of the breadboard, right above the button. The final circuit is shown in Figure X.

[picture of breadboard circuit]

Structural

After being assigned the laser cutting option and doing some research online, we quickly knew that we wanted to use wood; most of the projects we viewed online that involved assembly of parts were made out of wood and parts were fit together using wood joints. It made sense since, just based on life experience, wood has a much higher coefficient of friction compared to plastic and therefore wood joints would be much more secure. We did research on wood joints and decided to use mortise and tenon joints since they are relatively strong and require less complex cuts compared to other joints; many other joints require divots, grooves, or cuts that are parallel to the thin face of the wood, which are not feasible with laser cutting. We also decided to use the thickest option for wood (1/8” Balsa wood) since we felt that the tenon joints would be too susceptible to breaking if we used anything thinner.

Before beginning to create a CAD model, we arrived at a rough picture of what we wanted: (1) six faces to be cut out (one for each side of the breadboard), (2) the faces would be connected by mortise and tenon joints, (3) ‘holes’ would be left at the top and bottom end of the case for the USB cable and the LEDs and button, and (4) two pieces lodged into the sidewalls on top of the breadboard to keep it from moving up and down.

Regarding the dimensions, we modelled the dimensions of the case to perfectly fit the breadboard we modelled in Section 5 of Widget Lab 1. Although the measured dimensions of our physical breadboards did not match those of the one modelled in Widget Lab 1, we found that all dimensions of the latter were around 0.5 to 1 mm greater than the former. Therefore, by modelling a case to fit the slightly larger model, it provides a slight margin of error to ensure that the breadboard will fit in the case while not giving too much wiggle room. The final CAD model of the case is shown in Figure X.

* Overview of evaluation process (1-2 pages)
  + Description of final functionality level of systems prototype
    - How it works or why it does not work
  + Discuss test run and troubleshooting procedures followed
  + Discuss plan for checking functionality
  + Artefacts
    - MUST: picture of final prototype
    - Test results
    - Pictures from troubleshooting process

Plan for checking functionality

To check the functionality of the system, we referred to the *Objectives* and *System Requirements* sections (Section 5 and 6, respectively) of the assignment document. We determined that there were three main conditions to check for to ensure the functionality of the system: (1) the state of the LEDs changes when the button changes from *unpressed* to *pressed*, (2) the state of the LEDs change in a sequential and repetitive order (*off*, *plant-only*, *plant-and-worker*, *off*…), and (3) the correct LEDs light up for the intended state. Note that at the time this assignment was due, our case had not yet been fabricated and therefore structural requirements (such as the restriction of movement) were not able to be evaluated.

Testing and Troubleshooting

On the first test for functionality after finishing writing the code and building the circuit, neither of the LEDs were lighting up at all after pressing the button. The first thing I checked for is whether the error was in the code or the circuitry. To check this, I replaced the system code with the code from Section 12 of Widget Lab 1, which I knew worked, and added some code that would also light up the green LED when the button was pressed. Since I already knew the Widget Lab 1 code worked, if the LEDs did not light up as expected from Widget Lab 1, then I could deduce that the error was in the circuitry. But the LEDs lit up as expected when I pressed the button down, so I reasoned that the error was in the code. Surely enough, after opening the REPL window and running the system code again, there was an error message, shown in Figure X. Although not shown in Figure X, the line below stated that it was a syntax error. This was confusing to me since it seemed like the error was occurring in a file named “<stdin>”, but I was unable to find such a file. The most confusing part was that usually, Python prints out the line with the syntax error, but there was no hint as to what the syntax error was in this error message; it only stated that there was a syntax error. After looking online for people with similar experiences but with no success, I decided to retype the code line by line, inserting print statements occasionally, to find where it broke down. Oddly, I was able to type all the code without any errors appearing and the system worked as expected. I am still clueless as to what the reason for the error was, since if I had mistyped something in my code, it should’ve appeared as a syntax error in the *code.py* file.

Text

Description automatically generated with low confidence

Checking functionality of system

After the troubleshooting process, the final prototype, shown in Figure X, was completely functional and satisfied all three conditions listed in Section (plan for checking functionality). Beyond the expected pressing and releasing of the button, I tested two edge cases: (1) pressing and releasing the button very fast to test the chance that the button value changes mid-loop, and (2) holding the button down for a long time to make sure that the state changes only in response to *changes* in the button value and not the button value itself. The prototype passed both cases and is therefore functional regarding the code and circuitry components of the system.

* Explanation of individual contributions to partnership’s process, design and prototype (1-2 paragraph)

Both parties were heavily invested in each process of this assignment. In the design process, I was able to draw the parallel between the system and finite state machines we learned in ECE253, helping us both gain a clearer and more logical understanding of the system’s functionality. From there, developing the code became a very straightforward process for both of us. In the development process, I was more involved with the designing the structural component and creating the model in Fusion360. I thought of using wooden mortise and tenon joints to avoid the difficulties of laser cutting other types of joints and developed the CAD model submitted to MYFab. I also suggested mirroring the LED circuit we developed in Widget Lab 1 across the breadboard (mentioned in Section whatever it was) to help with the design’s aesthetic requirements. I was also able to help my partner identify a few syntax errors in their code in the troubleshooting process.

* Reflection on learning process (.5 to 1.5 pages)
  + What I learned from doing the lab
  + What was new
  + What helped my learning
  + Areas for growth and future learning related to this lab that interest me
  + How this experience related to some of my other academic and professional experiences
  + What I would do differently next time

I learned a great deal from this assignment since many of the technologies I worked with were new to me. While I had previously used AutoCAD before to develop 3D CAD models, this was my first time using Fusion360. This was also one of the few times I worked with a breadboard in my life. I worked with wires, resistors, and LEDs in high school before, but it was a long time ago and this was a fun refresher on how basic circuitry works on a breadboard. The technology that was newest to me and the one I learned the most about was microcontrollers. I had never worked with a Raspberry Pi before this lab, and while I’m certain that I’ve only scraped the surface of the capabilities of Raspberry Pis, it was still a fun and interesting process in learning about GPIOs, interpreting their specifications, using them to design LED circuits, and developing software that controls hardware through microcontrollers.

The Internet was a huge help in my learning throughout this lab. When encountering difficulties, errors, and doubts, the first source I consult is the Internet. The Internet was especially helpful to my learning in Widget Lab 1 when I was trying to select a current based on the GPIO specifications. Online forums and datasheets really helped clarify some concepts that I was confused about and also taught me a bit more about microcontroller GPIOs in general. Classmates also helped my learning a great deal. Although I used AutoCAD before, Fusion360 was a lot different and it was frustrating to use at times due to my unfamiliarity with the software. But friendly classmates guided me through the software and helped me familiarize myself, making it a much more enjoyable learning process, and I am thankful.

Areas related to this lab that I’m interested in learning more about definitely include the hardware components of the lab, as well as how they communicate with software. Learning about microcontrollers and building the circuit was the most enjoyable part of the lab for me, and the feasibility of creating cool devices out of these technologies that I can potentially use really excites me. My experience with this lab reminds me of numerous courses I’ve taken in the past. Besides the state diagram mentioned in Section WHATEVER IT WAS, the connection between software and hardware through the microcontroller also reminded me of the computer architecture portion of ECE253, where we worked with FPGAs and learned about using Verilog and Assembly to communicate with I/O devices. I was able to directly use my knowledge in Python learned from high school courses and ESC180 in the software component of the lab. Finally, I was able to draw a lot of connections between the circuitry and structural aspects of the lab to a design and technology course I took in high school. Similar to this lab, we built many small devices that used resistors, LEDs, and switches in that course. We also learned to use AutoCAD to model 3D objects, some of which we would end up physically constructing (such as a miniature toy car).

In the future, I will wait until I’m sure of which fabrication process I’m using before I start designing the structural components since the fabrication process has a huge impact on the design. In this lab, I wasted a lot of time designing unfeasible structures for laser cutting. Otherwise, I believe this lab went smoothly – I will learn from other mistakes I make in the future.

PART 2

* Widget Lab 1 Studio Report containing report on answers to questions from Widget Lab 1
* Widget Lab 1 section 10 and 11 – record some computation results, measurements, thoughts on circuit and coding work
* Look at section 9.4

1. My chosen resistor value was 330 ohms
2. Table

|  |  |  |
| --- | --- | --- |
|  | Expected | Measured |
| Current through LED | 5mA | 3.77mA |
| Voltage at GPIO pin before resistor | 3.1V | 3.12V |
| Voltage after resistor/before LED | 1.45V | 1.86V |
| Voltage after LED | 0V | 0V |

1. The LED is expected to light up whenever the button is pressed. The value of the LED is True when lit and False when not lit. On the other hand, due to the built-in pull-up resistor in the Raspberry Pi Pico, the value of the button is False when pressed and True when unpressed. Therefore the *not* in the last line allows a button value of True (unpressed) to correspond to an LED value of False (not lit), and a button value of False (pressed) to correspond to an LED value of True (lit).
2. While I used AutoCAD before, I familiarized myself with Fusion360 during the Widget Lab 1 studio sessions. I was also able to learn about conventional 3D printing and laser cutting preparation processes. Finally, I learned a lot about microcontrollers and their GPIOs when building the circuits.