Design Dossier Detailed Report

January 2024

1 Electrical Subsystem

1.1 Circuit Diagram

The circuit diagram 1 showcases the hardware configuration for a state-controlled lighting system, designed to correspond with the previously discussed state diagram for a multi-mode operational environment. It features a Raspberry Pi Pico micro-controller as the central processing unit, tasked with managing the system's modes based on user input via two push buttons. The three LEDs—red, green, and blue—represent the lighting states: 'System Off', 'Plant Mode', and 'Worker Mode'. The red and blue LEDs simulate the 'Plant Mode', optimized for plant photosynthesis, while the addition of the green LED indicates the 'Worker Mode', providing a spectrum conducive to human visibility. The push buttons are wired to GPIO pins on the microcontroller, and their state changes trigger the transitions between modes as defined by the state transition diagram. One button is designated as the 'System button', cycling the system through the modes, while the other, the 'Worker button', enables a direct transition from 'Plant Mode' to 'Worker Mode'. Current-limiting resistors ensure the protection and longevity of each LED, and common ground lines consolidate the circuit's return paths to the microcontroller. The circuit's design aligns with the state diagram's logic, providing a tangible interface for the described control architecture.

1.2 LED power needed to provide 10 DLI

The assignment handout specifies the metric for the energy received by the plants to be the Daily light integral.

The usual way to measure the Daily light integral requires the use of a Photosynthetically active radiation meter [1]. However, given the limited resources, we will approximate the Photosynthetically active radiation using other methods.

The minimum required Daily Light Integral must be must be greater than $10\frac{mol}{m^2 \cdot day}$. The first step will be converting this value to the Photosynthetic Photon Flux Density.

[1] Source: Quora question

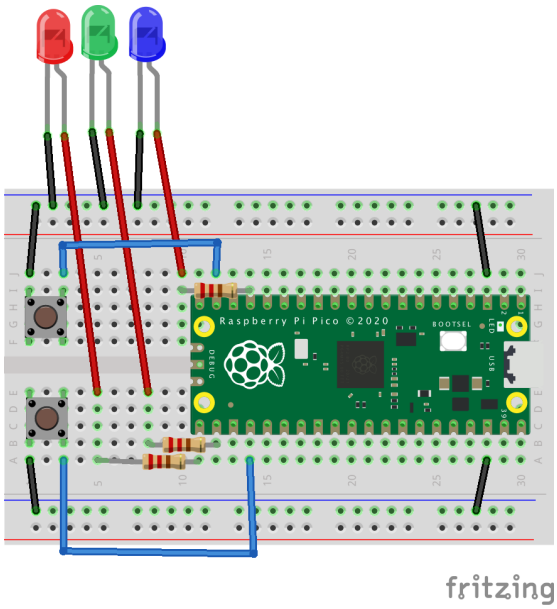


Figure 1: Multi-Mode Greenhouse Lighting Control Circuit

The calculation will be done according to the formula:
$DLI = 3.6 \cdot 10^{-3} \cdot PPFD \cdot \frac{Lighthours}{day}$ [2]
We assume the greenhouse LEDs will be turned on for the full day (24 hours). Substituting the values in and solving we obtain:

$$10\frac{mol}{m^2 \cdot day} = 3.6 \cdot 10^{-3} \cdot PPFD \cdot 24 \rightarrow PPFD = 115.74\,\frac{\mu mol}{m^2 s}$$

.

Now, in order to determine the required LED power, we will need to determine the number of photons the found PPFD corresponds to.

In order to make the correct calculation, we will make a number of approximations and assumptions:

- The plants are located a distance of 10 cm from the LEDs.
- The area covered by plants is circular, and is spanned by a 60-degree angle from the LED (as shown in Figure 4)
- We approximate the 3 LEDs in our design as one single LED.

First, we determine the area over which the plants are distributed:

$$A = 2 \cdot \pi \cdot R = 2 \cdot \pi \cdot 3 \cdot 10^{-2} \cdot sin(60°) = 0.16m^2$$

[2]Source

Now we can determine the number of photons per second needed for the plants:

$$n_{photons} = 115.74 \, \frac{\mu\text{mol}}{\text{m}^2\text{s}} \cdot 0.16m^2 \cdot \frac{6.022 \cdot 10^{17}}{1 \, \mu\text{mol}} = 1.115 \cdot 10^{19} \cdot \frac{1}{s}$$

Now we obtain the Power of these photons. First, we need to determine the energy of a single photon.

$$E = h \cdot f$$

where E is the energy of the photon, h is Planck's constant ($6.626... \cdot 10^{-34} J \cdot s$), and f is the frequency of the light. The frequency f can be related to the wavelength λ of the light by:

$$f = \frac{c}{\lambda}$$

Where c is the speed of light $3 \cdot 10^8 \frac{m}{s}$. The optimal light source falls within the blue range (425-450 nm) and red range (600 - 700). Let's assume the wavelength λ for the LEDs is around 550 nm. The frequency of the light:

$$f = \frac{3 \cdot 10^8}{660 \cdot 10^{-9}} = 4.54 \cdot 10^{14} Hz$$

The energy of a single photon:

$$E = h \cdot f = 3.01 \cdot 10^{-19} J$$

The power of these LEDs then be:

$$P = E \cdot n_{photons} = 3.35W$$

Additionally, we considered that only 120 Degrees of the LED contribute towards providing energy for the plants. If we consider that the LED fires photons for 180 Degrees, we can calculate the corrected power needed by the plants by integrating the corresponding surface area over the sphere, as seen in Figure 2.

$$P_{corrected} = \frac{2 \cdot ((3 + 3\sqrt{3}) * 10^{-2})^2 \cdot \pi}{2 \cdot \pi \cdot (3 + 3\sqrt{3}) * 10^{-2} \cdot 3 \cdot 10^{-2}} \cdot P = 9.15W$$

It is crucial to note that LEDs are not 100% efficient; they convert a portion of electrical power into light and the rest into heat. We approximate the efficency of the LEDs to be 81% (Source).

$$P_{corrected^2} = \frac{1}{0.81} \cdot P_{corrected} = 11.3W$$

. This gives that the power required per led is around 3.77 W.

This shows two important characteristics for a future final design:

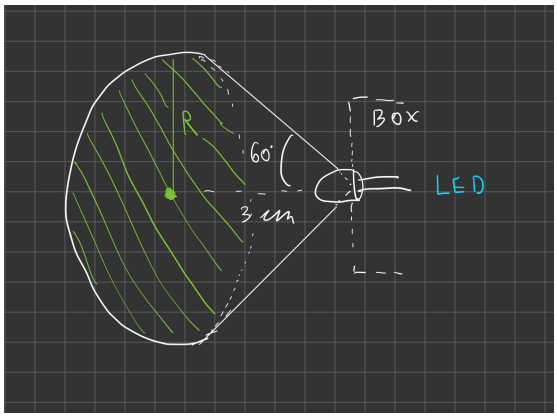


Figure 2: Diagram representing the correction accounting for the difference of surface area for the LED.

Considerations for final design

- If 3 leds are still used in the final design, the power of each one of them has to be at least 3.77W in order to meet the energy requirments. One example of a commercial LED that would meet this requirment is this.
- A more powerful board is required to supply enough power to the LEDs. The Pico used in the prototypes can only source 50mA total, at a theoretical upper voltage limit of 3.3V (Source: Rasberry Pi Pico documentation). In the final design, multiple relay devices like this one Can be used to control the highier power LEDs that are needed.

However, it will still be useful to calculate the DLI that the prototype built provides:

In this case, we start our calculations at the power of the LED. In the next section, we will justify our choice for resistor. In this section, we specify the desired current to be 10mA, and the desired voltage drop through the resistor to be 2.8V. The corresponding power will be: $P = V \cdot I = 2.8V \cdot 10mA = 0.028W$. As argued before, we account for the corrections for LED efficency, and the share of the light that actually arrives at the plants to obtain:

$$P_{corrected} = 0.028W \cdot 0.81 \cdot \frac{2 \cdot \pi \cdot (3 + 3\sqrt{3}) * 10^{-2} \cdot 3 \cdot 10^{-2}}{2 \cdot ((3 + 3\sqrt{3}) * 10^{-2})^2 \cdot \pi} = 8.303 \cdot 10^{-3}$$

Now, converting this power to photons per second:

$$n = \frac{8.303 \cdot 10^{-3}}{h \cdot 4.54 \cdot 10^{14} Hz} = 2.76 \cdot 10^{16} \frac{1}{s}$$

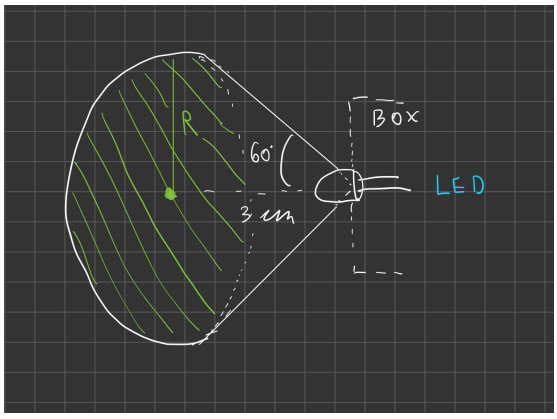


Figure 3: Diagram representing the area that an LED illuminates

$$PPFD = 2.76 \cdot 10^{16} \frac{1}{s} \cdot \frac{1\,\mu\text{mol}}{6.022 \cdot 10^{17}} \frac{1}{0.16m^2} = 0.289\,\frac{\mu\text{mol}}{\text{m}^2\text{s}}$$

Finally, we use the formula provided earlier to obtain the DLI:

$$DLI = 3.6 \cdot 10^{-3} \cdot PPFD \cdot 24 = 2.48 \cdot 10^{-2} \frac{mol}{m^2 \cdot day}$$

.

1.3 Resistor Calculation

1.3.1 Step 1: Determining upper and lower bounds for resistor

:

The RP2040 Dataset specifies that the current drawn across all pins cannot exceed 50 mA. Because we are using 3 LEDs (at least for the case when the workers are present, the maximum current that can be drawn per LED (pin) is:

$$I_{max} = \frac{50}{3} ma \approx 17ma$$

Additionally, we also need to make sure the voltage supplied by a pin does not go below 2.26V, which is the threshold for a logical high value for our Arduino. Looking at 4, we see that the maximum allowed current output will be also around 17 ma.

1.3.2 Minimum current output

Normally, in a the final design, the minimum current output would be determined according to the minimum luminous intensity. However, for this functional prototype we have already seen that the material constraints of this prototype do not allow to reach said intensity.

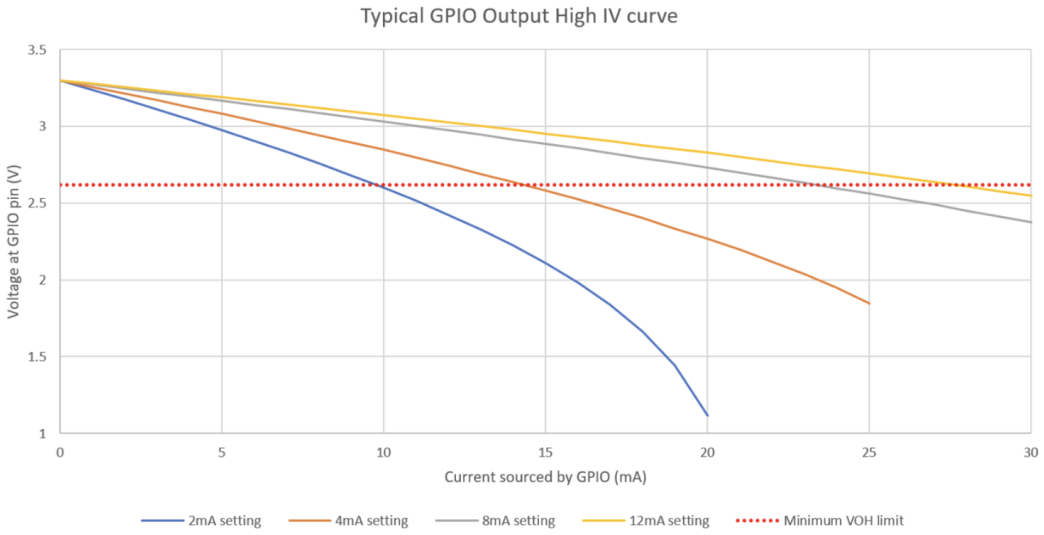


Figure 4: Voltage to Current output for RP2040 controlled board.

Therefore, we will concern ourselves with the minimum current for visibility of the LED (As the aim of the prototype is to demonstrate the functionality of the design to a person, who must be able to comfortably see the leds turning on and off). We refer to Figure 5. Experimentally we have determined that the minimum luminous intensity to be able to comfortably see the LED is around 0.5 of the reference value. This corresponds to a current value of 10mA.

1.4 Choosing optimal resistor

At this point we have determined the maximum and minimum current inputs per led for our application as:

- **Minimum:** 10 mA
- **Maximum:** 17 mA

However, can further attempt to meet the efficiency objective by providing an additional criteria for the choice of resistor. It is known that LEDs have a higher efficiency with lower currents, as this reduces the power that is lost in heat (Source: US department of energy). We will accordingly choose a resistor within the range that reduces the current in order to preserver as much of the efficiency as possible.

Another possible way to attempt to increase the efficiency of the LED would be to add a potentiometer to allow the user to change the resistance dynamically. However, there are two arguments based on which we have decided not to follow this path:

1. The user cannot be expected to perform electrical calculations in order to determine the most efficient resistance based on the condition in situ, this would be a violation of the easy to use objective.

2. In a greenhouse setting, energy is never waster. The more energy is generated by the LEDs, the more energy the plants will absorb, and the more value they will create for the greenhouse owner. source

We refer to the datasheet (Figure 6) again to obtain the voltage drop across the LED for the 10ma current as well as Figure 4 to determine the voltage supply on the pin for the given current, and we use the formula from a sparkfun article to obtain the ideal resistance:

$$R = \frac{2.8V - 1.8V}{0.01A} = 100\Omega$$

Although it is true that different LEDs can have different responses to voltage and current, after doing some research we decided to use 10mA as the threshold for the minimum intensity for all three LEDs. We were able to find some 5mm Green and Red LEDs in the Myfab scrap section, which allowed our LEDs to be all of the same size. We have also inspected the datasheets for the LEDs of these colors to confirm that 10 mA is a reasonable lower bound (although most datasheets did not provide explicit reference values.) (RED LED datasheet, Green LED datasheet)

Prototype Key decisions

- A 100Ω resistor will be used in the prototype, as this will allow for highest energy efficiency in our design, while still allowing users to comfortably see the functionality of our design.

2 Software Subsystem

2.1 Version control

The files used for the control of the microcontroler were backed up at every stage of its development on the GitHub platform. The version history contains all of the changes made to the file since its creation, and commens decribing what changes were made and why. The GitHub repository can be accessed at the following link: GitHub link.

2.2 Control Architecture

2.2.1 State Diagram

The state transition diagram 7 provided visualizes the control flow within the system. This consists of three states: ”System off”, ”Plant mode”, and ”Worker

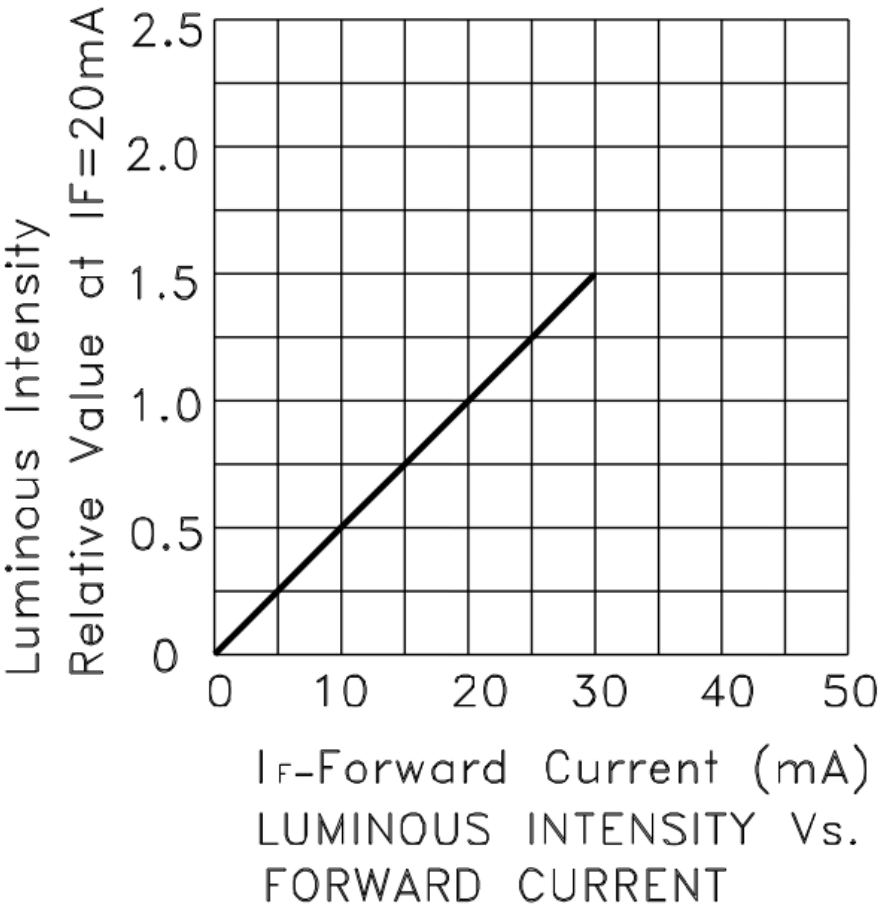


Figure 5: LED Current vs Intensity values from manufacturer data sheet

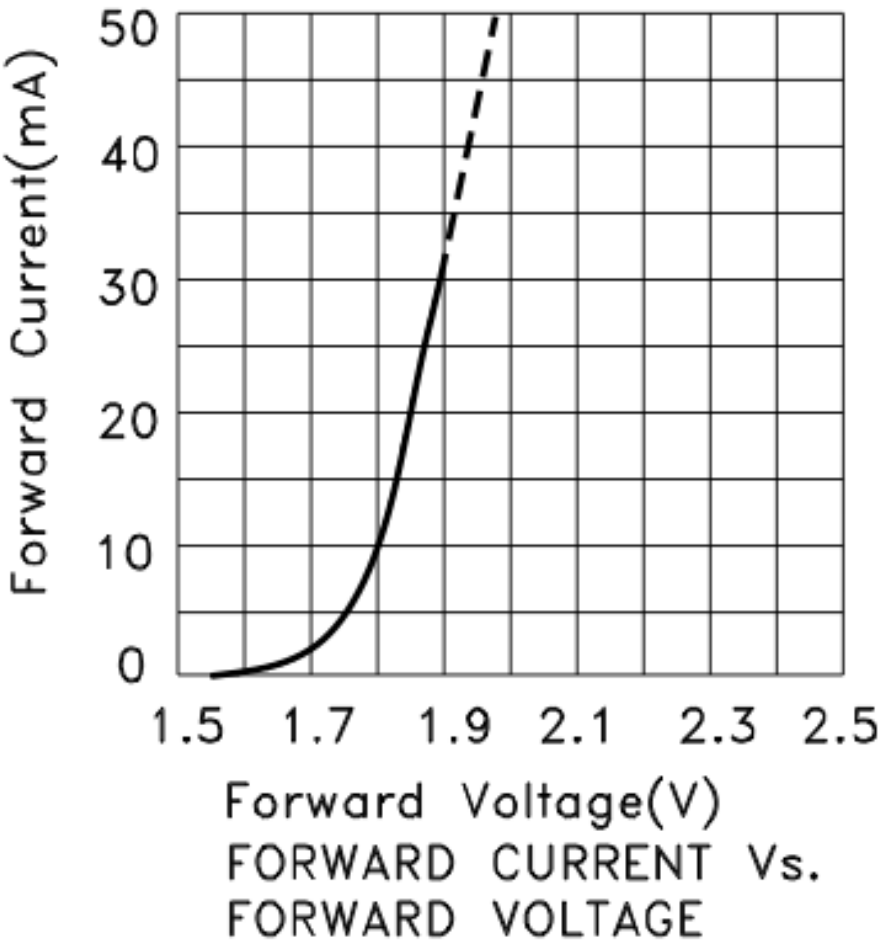


Figure 6: LED Current vs Voltage drop values from manufacturer data sheet

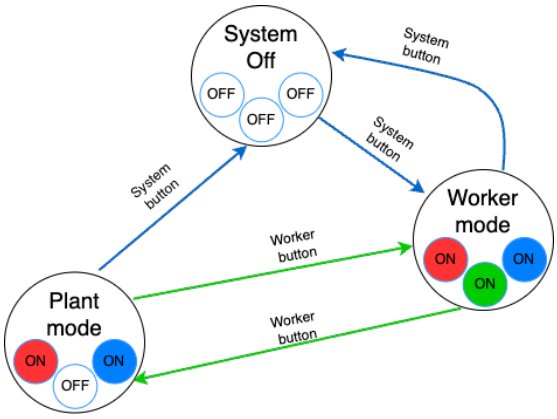


Figure 7: State diagram for our system.

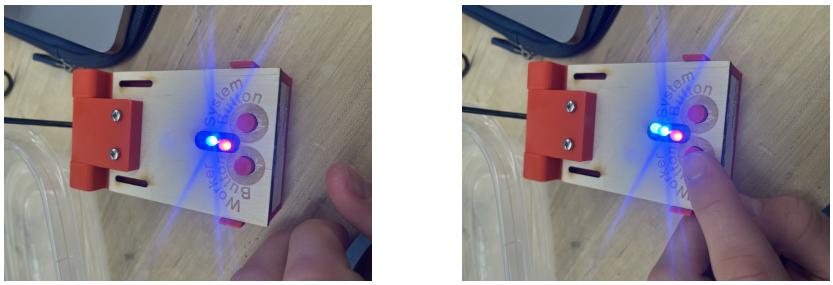
mode". In the "System Off" state, no lights are operational, conserving energy when illumination is not required. Transition from "System off" to "Worker mode" is achieved via a "System button", which when activated, illuminates all red, green, and blue LEDs suitable to formulate the white light to accommodate human visibility and work activities. A second press of the "System button" transitions the system back to "System off", and only transitions to "Plant mode" if the "Worker button" is pressed, where the green light will turn off, leaving red and blue light for plant growth. From this state, the worker can simply turn back to Worker mode by simply re-pressing the "Worker button". This control architecture allows for energy-efficient operation tailored to user presence and plant needs, with the state reverting to "System off" through the subsequent "System button".

2.3 Testing for the software subsystem

In order to ensure that the software subsystem works as expected, we put the system to the test in two ways:

Unit testing: We created a separate test.py file that replaces imported libraries that rely on being executed on the board by custom objects that simulate the state of the board. Then the code written for the board is tested using these objects, which allows us to inspect the state of the object (which simulated the board) at each moment.

Final testing: When the prototype was built and the board assembled, we tested the expected behaviour of the system by following all the possible combinations in the state diagram, and ensuring the correct LEDs lit up at each stage as shown in Figure 8



(a) Testing pink light (b) testing white light

Figure 8: Final testing of the software subsystem by inspecting if the correct color of LED is turned on on each stage of the sytem diagram.

2.4 Debugging stage

The debugging stage is a critical phase that follows software testing. It involves diagnosing and fixing the bugs identified, and unprecedentedly takes the most time. The primary goal is to enhance the stability and reliability of the software subsystem. The strategy involves:

1. Print Statement Utilization: Implement print statements to monitor the runtime behaviour of variables, function calls, and system states. This aids in pinpointing the exact location and nature of issues.

2. Problem Simplification: In cases of more complex bugs, reduce the problem domain by isolating the bug in a smaller, controlled environment. For example, we isolate the buttons or LEDs by implementing a simpler code targeting the specific button or LED via sample code under the Module's resources on Quercus. This approach facilitates easier identification and resolution of the issue.

3. Incremental testing approach: Adopt an incremental testing approach. This could be done through a thorough plan. For this project, we initially made the state diagrams (Figure 7) to exhibit the incremental steps of how we proceeded with the problem. This incremental approach ensures that at each stage of development, the functionality is confirmed, making it easier to pinpoint where issues arise when integrating the segments.

4. Version control utilization: Leverage version control systems like Git, or GitHub Desktop for a more friendly use. It allows reverting to previous stable versions if recent changes introduce bugs as well as peer review.

5. Peer Review: If issues persist, a peer review can provide new insights. A fresh perspective can uncover overlooked flaws, especially in the version control system like Github.

3 Structural Subsystem

The structural subsystem **??** is designed to encapsulate and interface with the control circuitry and the state transition logic. The housing provides a rigid enclosure for the Raspberry Pi Pico and the input/output components. It features interface elements, such as pushbuttons, that enable user interaction with the control system, directly reflecting the "System" and "Worker" buttons from the state diagram for mode selection.

The main body is fabricated from 3D-printed PLA, selected for its ease of use in manufacturing complex geometries. The top cover is constructed from plywood, providing an opportunity for precision lasercut holes and instructive text. Plywood offers a sturdy yet lightweight solution for the protective covering, and its use complements the plastic structure in terms of mechanical stability and overall durability. This combination of materials ensures that the enclosure is both economical and functional, with the capacity to house the Raspberry Pi Pico and facilitate user interactions via the pushbuttons, as dictated by the operational requirements of the state diagram.

3.1 Design Process

We decided that the structural element of the system would be the final subsystem to develop, molding the shape to fit around the other components. The main components of this system were the main body, the breadboard retention system, the closing mechanism, the securing mechanism, and the button extension. We went through a process of diverging on ideas for each of these mechanisms and then chose those that we decided best aligned with our given objectives.

3.2 Closing and Breadboard Retention Mechanisms

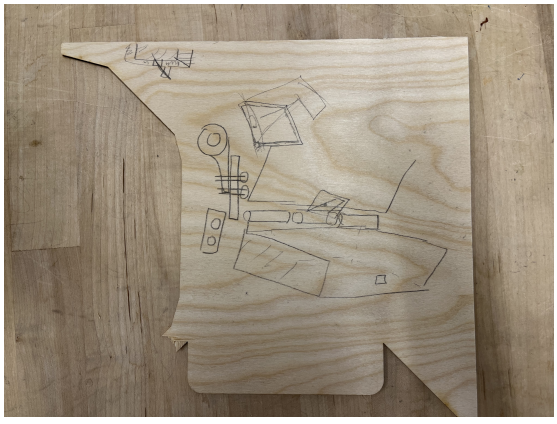


Figure 9: Hinge prototype drawings from Team Meeting.

After deciding the general shape and size of the box, the closing system and breadboard retention system needed to be flushed out. The three main design ideas were a push-fit lid, a hinging mechanism, or a screw-based system. These all had associated breadboard retention ideas with them: screwed in for the screw-based design and the push fit or slid in for the hinge design. The screw-based system would have the top and bottom sections secured together with screws. The push-fit lid would be laser-cut with cutouts to ensure security to the top 10. The hinge would involve a 3D-printed part that would connect the lower section of the box to a laser-cut lid 9.

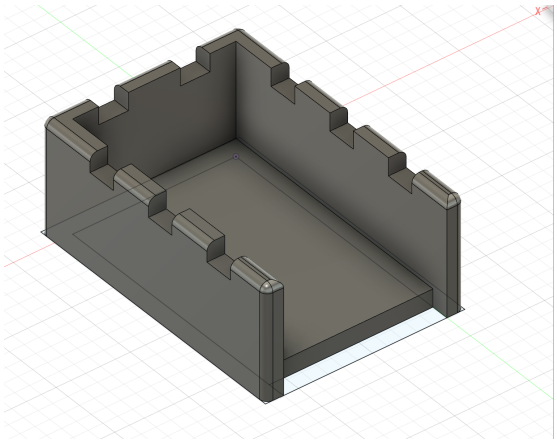


Figure 10: Push fit concept for closing mechanism.

The hinging mechanism was determined to be the most ideal idea. The requirements and specifications referenced for this decision were ease of use, the necessity of sufficient visibility, and the security of the breadboard. The hinge would allow the greatest ease of access and reassembly of the case compared to both other designs because it would stay assembled to the system instead of completely detaching 11. The LEDs would be the most visible if the lid could close over the top of them as opposed to sliding over them. As such, we designed the final height of the lid to allow for the LEDs to slightly poke out of the top. We also thought that the sliding breadboard retention system would provide the most ease of access for the breadboard while offering sufficient security.

3.3 Securing Mechanism For Hinged Lid

A hinged lid meant that it needed to be secured shut to ensure protection for the internal components. The main ideas for this were a traditional latch, a static 3D-printed latch, and a barring mechanism. Myfab does have a traditional latch, but after inspecting it, it appeared too large and obtrusive to be easy to use in the design. The 3D-printed static latch would be located on the top of the lower body of the box. It would be designed to be thin enough to bend easily so that that lid could temporarily move it out of the way, and would have a

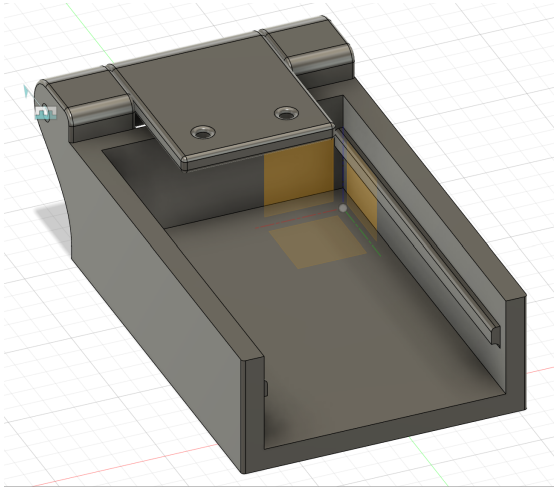


Figure 11: Push fit concept for closing mechanism.

small overhang that would wrap around the lid on top to secure it 12. The only worry with this design was the structural integrity of the latches as 3D-printed materials can be weak to shear forces, however, we figured that in a higher fidelity follow-up design, 3D-printed parts would likely be replaced by injection-molded parts which do not share the same weakness. The barred mechanism would utilize two slots on either side and the bar would slide through to secure the lid 13.it would be secured in the slots by a long slot cut within the bar and an extrusion within the containing slot. The static latch idea was determined to be the easiest to use as it did not introduce any more parts to the system and did not require any movement.

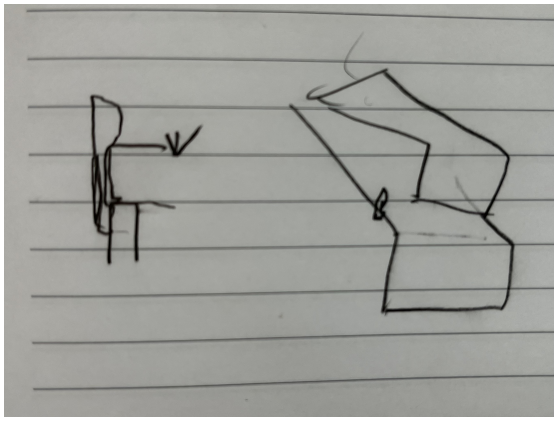


Figure 12: Conceptual Drawing of Static Latch Mechanism.

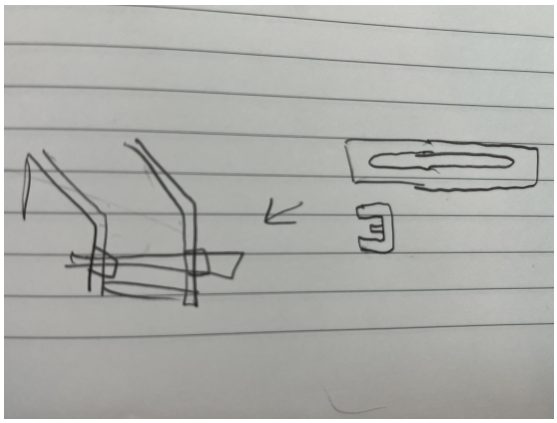


Figure 13: Conceptual Drawing of Barred Latch Mechanism.

3.4 Button-pusher

The spacing between the lid and the button in the main assembly informed the button-making process. Direct access to the button on the breadboard was nearly immediately shut down due to the desire to keep all electrical components contained within the body. This design process for this component was an iteration-based method as the general idea of a button-pusher was decided quickly. A drawing of the design shows the two components of the prototype: a main button pusher which has a ridge that sits under the lid to keep it contained as well as a lower housing that would be attached to the lid to keep the button in place when the lid is raised 14.

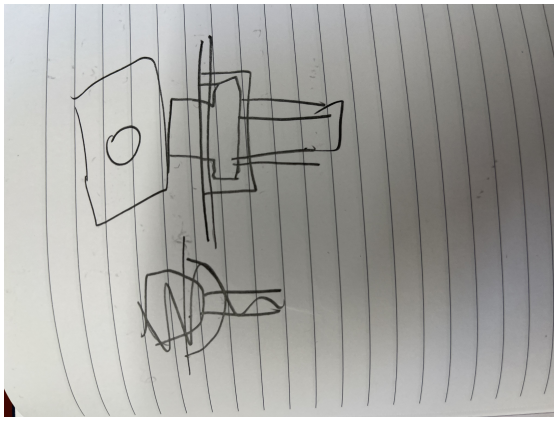


Figure 14: Conceptual design for a button mechanism.

3.5 Material Selection

The two options of material for this project were constrained to be plywood or 3D printed material. Due to the inclusion of multiple 3D-printed parts connected to the bottom body, we decided that it should be one contiguous 3D-printed piece to ensure manufacturing precision and structural integrity of the box. In addition, due to both the necessity to cut out holes from the top and the desire to print instructions on the top of the lid, we decided that the lid should be laser-cut.

3.6 System Integration

Hole locations in the laser-cut part for the button-pushers were determined based on the geometry of the breadboard buttons and LEDs from the main assembly in CAD. One notable integration decision that we had to make was on the orientation of the breadboard within the case. The first iterations had the plug-in adapter cutout facing towards the front of the assembly. Once we put the circuit design together with the structural design into an assembly, we recognized that our planned button locations would overlap with our hinging mechanism. Because of this, we had to flip the orientation of the breadboard within the structural element and accordingly relocate the cable inlet location to the back of the case. This had compounded effects and we then had to alter the hinge to be higher up to allow access to the cable inlet in the back.