ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT (DRAFT)

STM32-Powered Physical Model of Network Flow

Team #17

BOLIN ZHANG
(bolinz3@illinois.edu)
JIAHAO FANG
(jiahaof3@illinois.edu)
YIYANG HUANG
(yiyangh5@illinois.edu)
ZIYUAN CHEN
(ziyuanc3@illinois.edu)

Sponsor: Pavel Loskot

TA: Yue Yu

May 10, 2024

Contents

1 Introduction

1.1 Problem Statement

Many real-world systems involve flows over networks. Logistic systems, transportation networks, and the Internet are all carefully designed to meet the capacity and cost requirements. However, in algorithm courses, flow optimization problems can be hard to imagine. Students often struggle to quantitatively predict how each tune in the constraints will affect the optimal solution using mere intuition.

Having a physical model can provide a more intuitive sense of "tuning" the network by assigning a knob to each parameter and a strip of LEDs to each link. Such a model also has the potential to *dynamically* visualize more complex scenarios in realistic flow management, such as the presence of routing hubs, congestion, and packet delay.

1.2 Solution Overview

Our team aims to build a *modular, reconfigurable* hardware emulator to visualize network flows under capacity constraints on links. Each node can be configured as a source, a sink, or a "transfer station" that holds zero flux. Solutions will be computed on a connected computer using the Ford-Fulkerson algorithm in Python. Display will be controlled using an embedded STM32 microcontroller.

The intermodular communication protocol has gone through several revisions. We moved from a MUX-driven protocol (first iteration) or direct access using Arduino (second iteration) and instead opted for a more flexible peer-to-peer (P2P) communication model (third iteration) where only one node maintains a direct UART connection with the computer, and identical "configuration info" strings are distributed for local processing.

Specifically, in the final layout, we plan to simulate a simple yet realistic network topology with "forwarding hubs," incorporating 6 nodes and 9 links in total. We hope this toolset will provide an intuitive visual aid and facilitate the understanding of flow algorithms in a classroom setting, especially where the network in discussion is inherently dynamic (e.g., routing packets in the Internet).

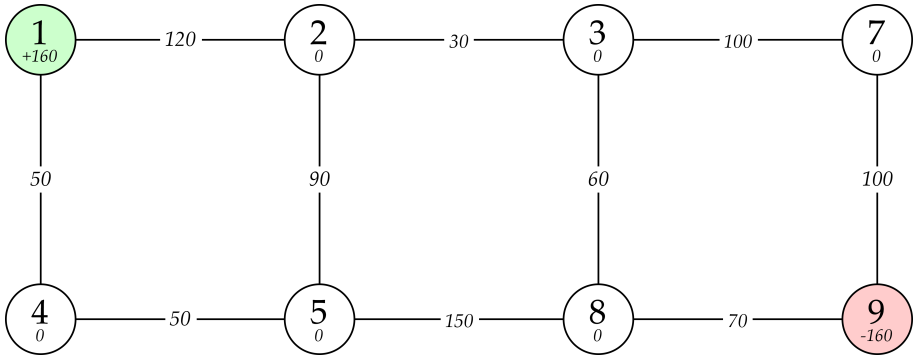

Figure 1: The network topology to be implemented.

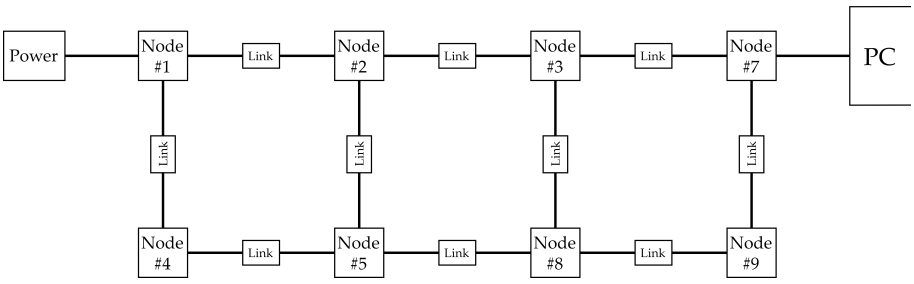

Figure 2: Layout of intermodular connections.

Meanwhile, we are making several assumptions that simplify the problem:

- This is a small-scale network with up to 4 links per node – that is, if the network is ever to be reconfigured.
- The nodes don't have buffers and is able to respond instantaneously to changes in capacity constraints.
- All links are bidirectional, and both directions have the same capacity.

1.3 High-Level Requirements

- The physical model should be modular, i.e., each node has a number of "slots" reserved for installing new links. We aim to serve 6 partially connected nodes.
- The software should communicate with all nodes and pipes and update the flows in real-time (within 500ms) in response to changes in setup.
- The algorithm should handle and report edge cases such as a network with zero or multiple feasible flows.

2 Design

In our physical model, a complex network is abstracted into a series of pipes, representing the communication **links** characterized by their designated capacities ("link capacity"). The embedded LEDs depict the dynamic flow ("link flux") of data packets coursing through the network.

Each **node** within this network, also represented by a dedicated PCB, is associated with a "node flux" value, offering the flexibility to characterize each node as a source with outgoing flow, a sink with incoming flow, or a neutral transfer station with no net flow. This modular approach underpins the system's design, ensuring adaptability and ease of modification as the network model evolves. (We considered integrating potentiometers (knobs) onto the PCBs, but soon realized that space is insufficient.)

The transition to the STM32 microcontroller platform on each PCB has substantially increased the system's capabilities. Tasked with collecting network configuration, the microcontroller reads from the network configuration string, control the LEDs correspondingly, and forwards it to the next neighbor.

The Python code running on an external computer implements the Ford-Fulkerson algorithm that computes network flows while considering all constraints (node flux and link capacity). A software GUI displays the solution alongside the physical model due to limited space (number of LEDs and pins for interconnection) in each node and link.

The following diagram details the UART interface between modules. The links are *not* directly connected to the computer, and thus the capacity values configured must be passed indirectly through the nodes.

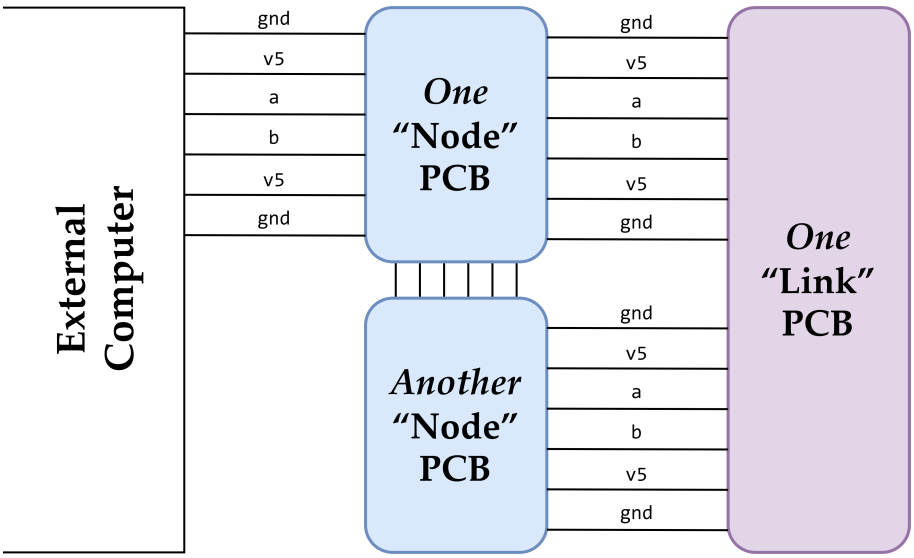

Figure 3: Intermodular communication interface signals.

2.1 "Node" PCBs

We now investigate the circuit components and finite-state machine design in the modules, except for the external computer which is primarily concerned with the software subsystem. Circuit diagrams are also provided wherever necessary.

Each node is a customized PCB board that includes

- one STM32 **microcontroller** that drives the display and signals,
- sixteen **LEDs** that indicates the "node flux", and
- four groups of UART serial **interfaces** with link PCBs, the computer, or power.

The Node PCB forms a critical part of our toolbox, serving as the visualization point for network traffic at each node within the system. Central to the Node PCB is the integration of the `74hc595` series shift registers. These components enable the expansion of output ports through serial-to-parallel conversion, allowing the microcontroller to control a larger array of LEDs while utilizing fewer I/O pins.

In the schematic, the connection of resistors `R1` to `R22` in series with the LEDs facilitates the display of network traffic flow. Each resistor-LED pair acts as an individual indicator. The binary states managed by the shift register correspond to the on or off states of the LEDs, effectively demonstrating the flow through each node.

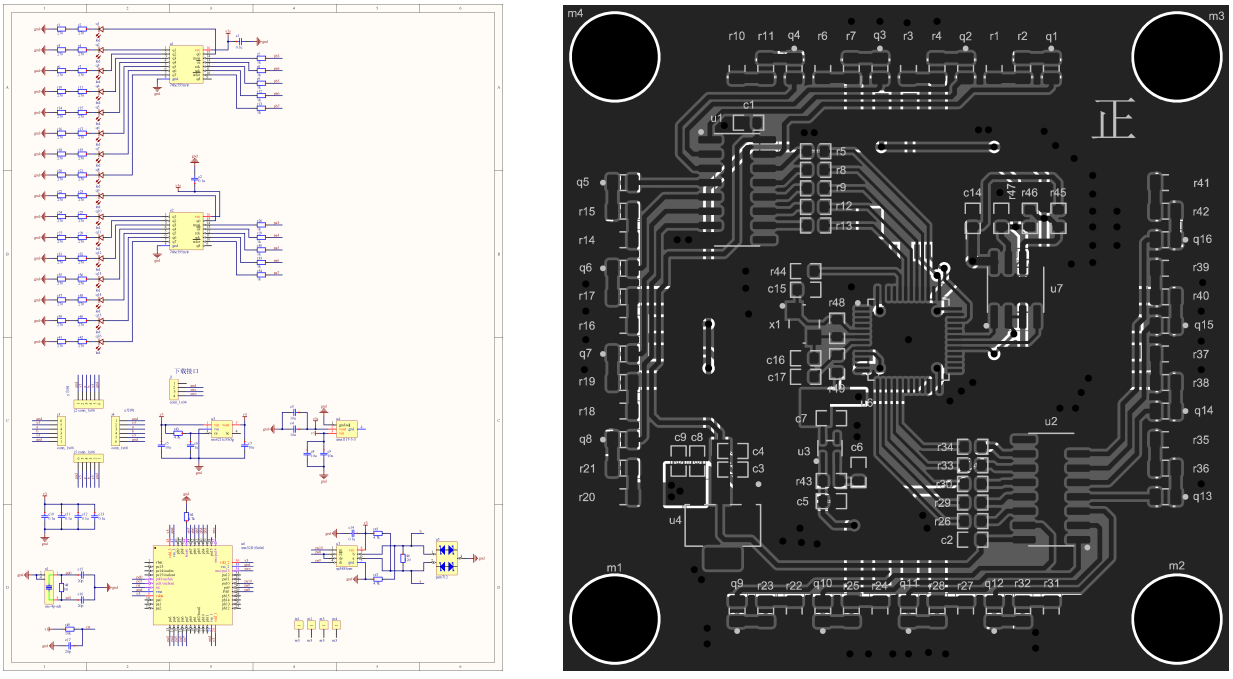

Figure 4: Circuit schematic and PCB layout for the nodes

2.2 “Link” PCBs

We considered treating the links as mere LED strips and leave all capacity configurations to the node PCBs, but this appears counterintuitive and would require a complex communication protocol to set up the network topology. Therefore, each link is also a customized PCB board that includes

- one STM32 **microcontroller** that drives the display and signals,
- sixteen **LEDs** that indicates the “link flux” (exactly one LED is on at any time, and the “rolling” speed indicates the flux amount), and
- two groups of UART serial **interfaces** with node PCBs.

The Link PCBs function as conduits for demonstrating the flow of data between nodes, with LED arrays illustrating the link status and activity within the network.

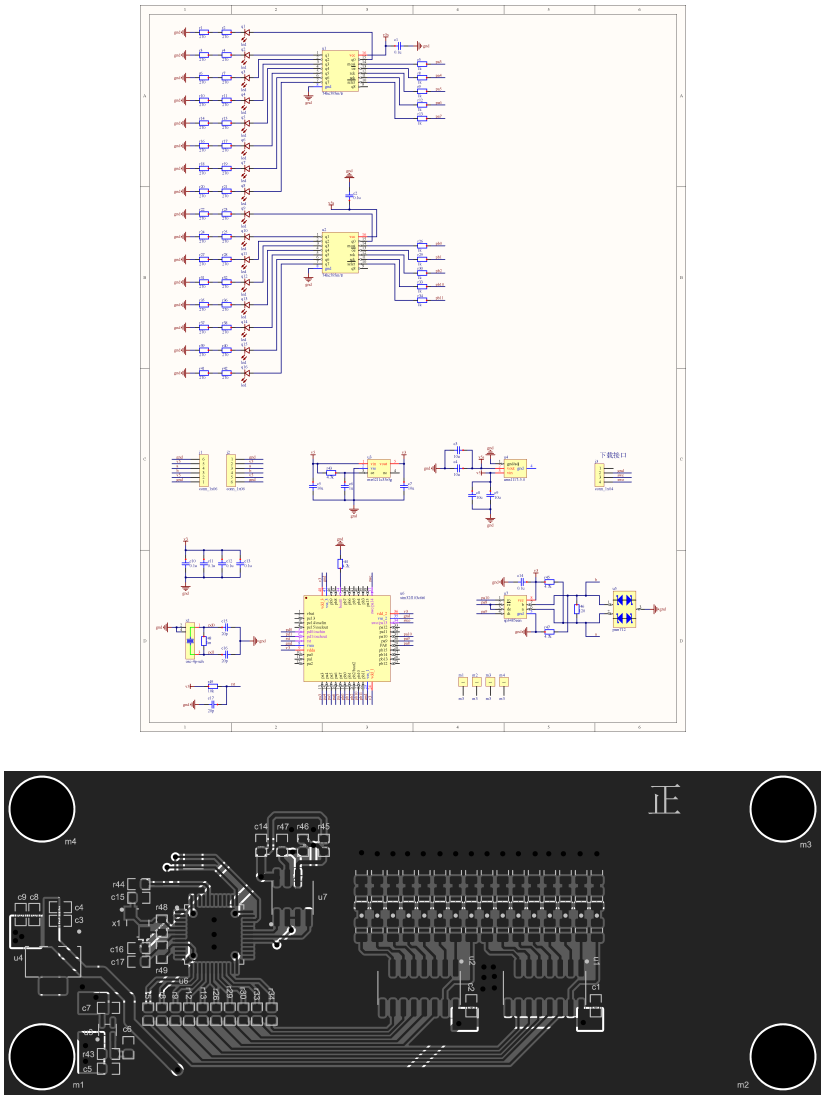

Figure 5: Circuit schematic and PCB layout for the links.

In the detailed schematic provided, the `74hc595` shift registers are used to control arrays of LEDs, each corresponding to a particular link state, illuminated to represent the presence of data flow. The shift registers enable the control of multiple LEDs while minimizing the GPIO usage of the controlling microcontroller. Each LED is connected in series with a resistor, which limits the current to prevent damage to the LEDs.

During operation, the shift register receives serial data from the microcontroller. Upon receiving a clock pulse, the register shifts this data through, setting each LED's state accordingly. Then a latch signal is applied, which updates the output pins and changes the LEDs' states simultaneously. This design allows for real-time updates to the network flow visualization without perceptible lag to the user.

2.3 Power PCB

We used to have a MUX PCB that (1) helps simplify the interface to computer while (2) supplying power, clock, and reset signals to all nodes and links. However, due to the exceedingly large number of I/O ports, this part of design has been refactored to a dedicated power PCB that supplies 5V power, ground, and clock signals to all PCBs.

Similar to the P2P communication network in which the protocol implies that only one node needs to be connected to the external computer, we design the power supply such that only one PCB needs to be connected to the power source.

Power management on the board is handled by the `MEC6211` voltage regulator, ensuring that the STM32 and other components receive a stable voltage, critical for maintaining reliable operation. Surrounding are various passive components like resistors and capacitors, which stabilize the signals and power supply. The capacitors closer to the power inputs, marked as `C4` through `C7`, are likely for decoupling purposes, filtering out noise from the power supply to the microcontroller and other sensitive components.

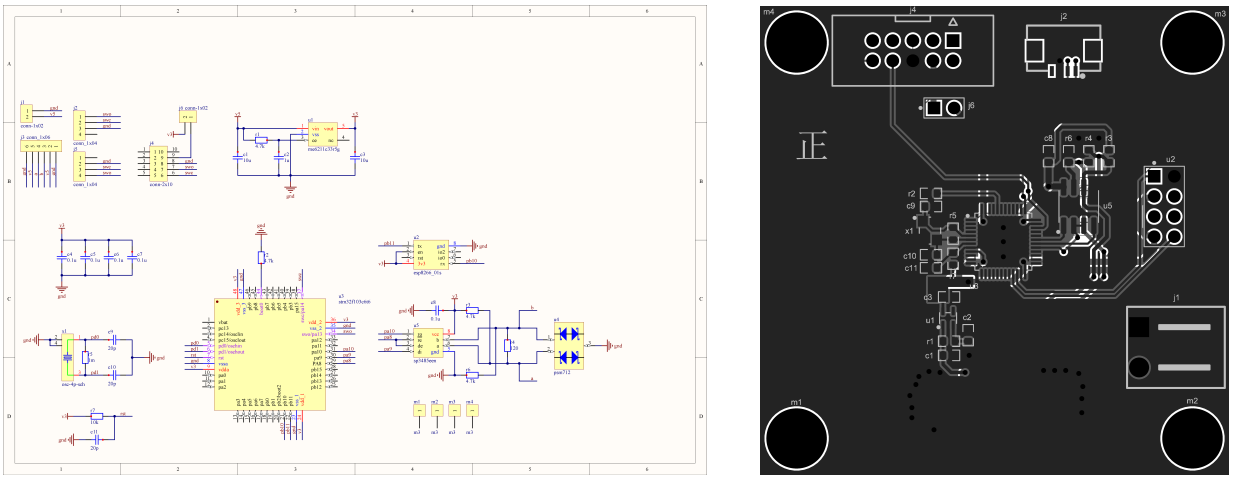

Figure 6: Circuit schematic and PCB layout for the power.

The communication lines between the STM32 and other peripherals are safeguarded by the `psm712` diodes, which protect against voltage spikes that could otherwise damage the microcontroller. There is also a crystal oscillator circuit, fundamental for providing a precise clock signal and ensuring accurate timing for all operations.

2.4 Intermodular Communication

In our project, the communication between the Python application and the physical hardware, specifically the printed circuit board (PCB), is facilitated through `PySerial`. This is a Python library that provides a convenient interface to the serial ports, enabling the sending and receiving of data over serial communication lines with minimal effort. The choice of `PySerial` is driven by its simplicity and robustness, allowing developers to manage serial communication through a few straightforward lines of code.

The underlying communication protocol employed in the project is RS485, known for its reliability and efficiency in enabling long-distance and high-speed data transmission. The SP3485EEN handles the electrical aspects of RS-485 communication [1], [2], ensuring robust data transmission even over long distances and in electrically noisy environments. The STM32 sends and receives data to/from the SP3485EEN. It controls the transceiver's operation by managing the DE/RE pins to switch between sending and receiving modes.

2.5 Software Controller

We incorporate the **Ford-Fulkerson algorithm** that starts with an initial flow of zero and repeatedly finds an augmenting path from the source to the sink within the residual graph using Breadth-First Search (BFS). The augmenting path is a path in the residual graph that has available capacity for increasing the flow. The algorithm increases the flow along an augmenting path until no further augmenting paths are available. The maximum flow is then the sum of the flows along all the paths from the source to the sink.

The Python software is expected to

- read the node configurations and link capacities from the GUI,
- computes the maximal flow using an optimized Ford-Fulkerson algorithm, and
- updates the flow display on each node and link in real-time.

2.6 Graphical User Interface

The graphical user interface (GUI) receives the flow solution and visualizes the network flow data. The design and development of the GUI are guided by user-centric principles to ensure intuitiveness, ease of use, and functionality. The key functions are:

- **Real-time flow visualization.** The GUI should display the real-time flow of data packets within the network, preferably assisted by *animated* changes in the network diagram, correlating with the actual flow of data through the nodes and links.
- **Algorithm control and monitoring.** The user should be able to initiate, pause, or stop the flow computation algorithm and monitor its progress. The GUI should provide a console or log view to observe the real-time output from the algorithm, including any alerts or error messages.
- **Error handling and feedback.** Prompt and clear feedback should be given for any invalid actions or errors in configuration, e.g., when there are no *or multiple* feasible solutions, or when the conservation principle is violated. This includes the visualization of flow states that are not permissible due to the current network setup.
- **Responsiveness and scalability.** The GUI should be responsive to different screen sizes and resolutions, ensuring usability across various devices.

The GUI will be developed in a modular fashion, allowing for future enhancements and features to be added with minimal disruption to the existing system. By focusing on these core functions, the GUI will facilitate an effective and educational experience.

2.7 Outer Packaging

The entire toolbox will *reside on a vertical surface* for convenient display on whiteboards. We aim to make the outer packaging structure and overall appearance of the toolbox both aesthetically pleasing and functional. The following considerations guide the design of our product's exterior:

- **Acrylic casing.** The node and link PCBs will be encased in high-quality acrylic panels, allowing for the visibility of the internal components and LED indicators.
- **LED indicators.** The flow of data through the network will be represented by LED lights housed in clear, durable tubes that not only protect the electronics but also distribute light evenly, making the flow visually discernible from all angles.
- **Modularity and expandability.** The modular design will allow for the network to be expanded or reconfigured. This includes detachable nodes and links, which can be securely attached or removed without the need for specialized tools. *Note that the interface has a large number of signals, which may require a revision for over 6 nodes.*
- **Environmentally conscious.** The design process will incorporate environmentally friendly materials and practices, including recyclable plastics and efficient LED lighting, to minimize the ecological footprint of our product.

3 Reliability and Verifiability

3.1 Requirements and Verification

Component	Requirement	Verification
Knobs	The embedded ADC should map the position of the knob potentiometer to an 8-bit signed integer.	1. Power on a node PCB. *We don't power on the knob independently to allow data preprocessing by the microcontroller.* 2. Slowly turn the knob from the minimum position to the maximum position. 3. Observe that the displayed flux values increase uniformly from -127 to 127.
"Node" PCBs	The node PCB should respond to a "read NODE" command within 1 clock cycle.	1. Power on a node PCB. 2. Simultaneously monitor its clock, `read`, and `resp` signals on an oscilloscope. 3. Drive `11` ("NODE") on the `cmd` bus and assert `read`. 4. Observe that `resp` is asserted within 1 clock cycle and `rdata` matches the segment display.
"Link" PCBs	The link PCB should respond to a "read" command within 1 clock cycle.	1. Power on a link PCB. 2. Simultaneously monitor its clock, `read`, and `resp` signals on an oscilloscope. 3. Assert the `read` signal. 4. Observe that `resp` is asserted within 1 clock cycle and `rdata` matches the segment display.

Table 1: Technical requirements and verification procedures (Part 1).

Component	Requirement	Verification
Algorithm	The algorithm should respond to changes in the network setup within 500ms.	1. Power on the system. 2. Set the source and sink by deviating some node knobs from the "neutral" position. Ensure the displayed node flux values sum to zero. 3. Enable all links except for leaving *the last leap* on a feasible path at zero capacity. 4. Enable the "bottleneck" and observe that the flow is redistributed within 500ms.
GUI	The system should report an error when no feasible flow exists.	1. Power on the system. 2. Set all node and link knobs to the neutral position except for one single source and a matching sink. 3. Turn on all links except for all links entering the sink. 4. Observe that the LEDs display no flow while the GUI reports an unsolvable case.
GUI	The system should report an error when the flow is not conserved.	1. Starting from the last experiment, slightly tune the sink flux to unmatch the source. 2. Observe that the LEDs display no flow while the GUI reports a flow imbalance.
Packaging	The system should attach firmly to a magnetic whiteboard and be visible from a distance of 3 meters.	1. Configure the network such that the solution uses over 90% capacity for all links. 2. Observe that all LEDs and segment digits are clearly visible from 3 meters away.

Table 2: Technical requirements and verification procedures (Part 2).

3.2 Tolerance Analysis

- (Node and Link PCBs) Tolerance of Resistors and Capacitors
  - Impact: They affect the timing and signal shaping within the node's circuitry, possibly leading to misinterpretation of signals or timing mismatches.
  - Analysis: Consider how the RC time constants change due to variations in resistor and capacitor values, and how they could affect the signal levels and timing, especially for signals interfacing with the MUX and links.
  - Simulation: We may use RC response curves to illustrate how different RC time constants affect the rise and fall times of the signal waveforms.
- (Node and Link PCBs) Accuracy of Knob Potentiometers
  - Impact: It affects the precision with which node flux can be set, potentially leading to inaccurate flow visualization.
  - Analysis: Determine the range of actual values for a supposed set value and how this affects the node's status as a source, sink, or transfer station.
  - Simulation: We may display a range of actual flux values corresponding to a set position on the knob to highlight the variability due to tolerance.
- (Link PCBs) Variability of LED Display
  - Impact: Variability in LED brightness and color could lead to inconsistent flow visualization across different links.
  - Analysis: Consider the variation in LED brightness and color due to current and voltage tolerances and its impact on visual accuracy.
  - Simulation: We may use a comparison chart to show the expected brightness range of LEDs under different current and voltage conditions due to tolerances.
- (Intermodular Protocol) Voltage Levels and Noise Margins
  - Impact: Fluctuations in voltage levels can affect the logic levels interpreted by the microcontrollers and introduce errors in the protocol.
  - Analysis: Examine how variations in voltage levels (due to power supply tolerances or signal integrity issues) might impact the detection of high and low states in the communication protocol.
  - Simulation: We may show the acceptable voltage levels for logic `0` and logic `1` for the microcontrollers and how variations might lead to incorrect logic level interpretation.

4 Cost Analysis

Estimated labor costs [3]: 4 people$\times 10$ hrs/week$\times 8$ weeks$\times$RMB 200/hr $=$ **RMB 64,000**

Component	Quantity	Cost
Magnetic Isolation USB 485 Bidirectional Serial Converter	1	RMB 59.00
Chip Resistor	1 set	RMB 25.00
Switching Mode Power Supply	1	RMB 51.00
High-power Two Pin Plug	2	RMB 7.20
Spacing Plug-in Terminal Block	3	RMB 1.92
Curved Pin Socket	3	RMB 0.36
FFC/FPC Connector	30	RMB 8.40
2.54mm PIN	3	RMB 6.60
FC Double Headed Ribbon Cable	3	RMB 7.20
ST - LINK V2 STM 32	1	RMB 19.80
High Brightness SMT LED	500	RMB 11.00
XH 2.54 Terminal Wire	4	RMB 90.00
XH 6P Recumbent Patch	1	RMB 30.48
PCB Source	5	RMB 46.60
PCB Node	20	RMB 58.92
PCB Link	20	RMB 63.07
AMS1117	50	RMB 12.74
LDO	50	RMB 22.10
TVS/ESD	50	RMB 15.15
RS-485	45	RMB 66.35
Shift Register	85	RMB 38.16
304 Stainless Steel Bolt Set M5*70	1 set	RMB 6.9
304 Stainless Steel Bolt Set M5*60	1 set	RMB 6.9
304 Stainless Steel Nut M5	50	RMB 2.9
M3 Diamond Pointed Round File	1	RMB 11.00
Chevrolet Board 1200*450	1	RMB 60.00
Acrylic Board 1200*450	2	RMB 260
Cable ties	1000	RMB 5.36
Software Copyright Fee	/	RMB 580.00
Total		**RMB 1574.11**

Table 3: Cost breakdown of all circuit components.

5 Conclusion

5.1 Ethical Concerns

To avoid ethical breaches in the development and deployment of our toolbox, we commit to adhering closely to the principles outlined in both the IEEE Code of Ethics [4] and the ACM Code of Ethics [5]. Key considerations include but are not limited to

- Respecting *intellectual honesty* (ACM, Clause 1.5), acknowledging contributions accurately, adopt secure coding practices, and avoiding plagiarism in development.
- Committing to *inclusivity and accessibility* (ACM, Clause 1.4). For example, both the model and the GUI should be designed to be usable by a broad spectrum of individuals and accommodate users with diverse technical backgrounds.
- Supporting *sustainable development* (ACM, Clause 3.4; IEEE, Clause 1). This includes choosing recyclable and sustainable materials for hardware components and designing the embedded electrical system for energy efficiency.
- Mitigating the *risk of overreliance* by positioning our tool as a supplementary, instead of replacement, of traditional educational resources, in compliance with the IEEE's commitment to continuous learning (IEEE, Clause 6).

By fostering an environment of transparency, responsibility, and respect for user rights, we aim to not only comply with professional ethical standards but also contribute positively to the educational and technological communities.

5.2 Safety Concerns

This project involves the use of diodes, microcontrollers, and light bulbs to simulate the network information transmission flow. Recognizing the associated electrical, fire, mechanical, chemical, and operational hazards both in the development and deployment stages, we will abide by the IEEE National Electrical Safety Code [6] through

- Implementing comprehensive safety measures including protection against *electric* shocks (e.g., insulated tools) and *fire* precautions (e.g., circuit breakers, fuses) to prevent overheating and short circuits.
- Securely mounting all PCB board components to ensure *mechanical* robustness.
- Using protective gears during assembly involving batteries and soldering operations, and safely disposing hazardous *chemical* waste.
- Ensuring the safety of users (ACM, Clause 2.9; IEEE, Clause 1) by rigorously testing the system to prevent any *operational* hazards. For instance, the number of small parts in the physical model should be minimized to prevent choking hazards.
- Training in safe handling practices and emergency procedures.

References

[1] Texas Instruments. "The RS-485 Design Guide." (2021), [Online]. Available: https://ecs.grainger.illinois.edu/student-resources/offers/salary (visited on 05/05/2024).

[2] Analog Devices. "RS485 Quick Guide." (), [Online]. Available: https://www.analog.com/media/en/technical-documentation/product-selector-card/rs485fe.pdf (visited on 05/05/2024).

[3] Grainger Engineering Career Services. "2021-2022 Graduates Salary Data." (2022), [Online]. Available: https://ecs.grainger.illinois.edu/student-resources/offers/salary (visited on 03/27/2024).

[4] IEEE. "IEEE Code of Ethics." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 03/06/2024).

[5] ACM. "ACM Code of Ethics." (2018), [Online]. Available: https://www.acm.org/code-of-ethics (visited on 03/06/2024).

[6] "2023 National Electrical Safety Code® (NESC®)," *2023 National Electrical Safety Code(R) (NESC(R))*, pp. 1–365, 2022. DOI: 10.1109/IEEESTD.2022.9825487.