Implementing the Atanasoff-Berry Computer With Modern Technology

Design Document

sdmay25-29

Dr. Alexander Stoytchev

Connor Hand / Client Interaction and Team Organization

Zach Scurlock / Testing and Individual Component Design

Noah Butler / Testing and Individual Component Design

Peter Hurd / Testing and Individual Component Design / Budget Handling

William Mayer / Meeting Tracking and Note-Taking

sdmay25-29@iastate.edu

https://sdmay25-29.sd.ece.iastate.edu/

Executive Summary

The Atanasoff-Berry Computer (ABC) was the world's first electronic digital computer, a milestone in computational history. Our project seeks to recreate the ABC using modern technology to preserve its educational and historical significance. The goal is to create a hands-on, interactive tool that aids educators and students in understanding foundational computing principles while honoring the historical importance of the original ABC. By providing a tangible example of early computational logic, this project enhances educational outcomes and inspires a deeper appreciation for the evolution of technology.

This project addresses the need for accessible tools that help bridge the gap between theoretical concepts and practical understanding in computer engineering. Our recreated ABC must accurately replicate the computational logic of the original while remaining user-friendly and interactive. To achieve this, we are using integrated circuits (ICs) and breadboards while ensuring our design stays true to the spirit of the original machine. Input will be managed through an Android-based interface, emulating the ABC's punch-card system while improving accessibility. Core subsystems, such as the add-subtract mechanism, base converter, memory drums, and control system, are designed to function with modern components, yet follow the logic of the original machine.

We have developed all the components of the ABC on breadboards. The breadboards were designed modularly such that components can be removed and tested individually. We opted to place our breadboards onto a sheet of plywood, using velcro, to have a consistent layout and be able to easily transport our computer. The Android-based interface is complete and fully functional. Our Android tablets communicate with ESP32s to control the input/output of the machine. The tablets are mounted to the plywood using 3D-printed mounts.

Our design successfully balances historical accuracy with modern practicality, ensuring it meets the needs of students, educators, and hobbyists. Our design utilizes LEDs to display calculations at each step of the computing process, furthering the ease of understanding for students. Our recreation not only revives the ABC's legacy but also contributes to advancing the education of future generations of computer engineers.

Learning Summary

Development Standards & Practices Used

- Circuit Practices:
    - Breadboarding for initial prototyping of circuits.
    - Use of ICs for implementing digital logic.
    - Modular breadboard design for final module assembly.
    - Adherence to proper grounding and signal integrity techniques during circuit design.
- Hardware Practices:
    - Sourcing reliable and cost-effective components that are compliant with industry standards.
    - Modular design to simplify testing and integration.
    - Compact design to fit within a typical educational workspace.
- Software Practices:
    - Use of version control for code and documentation.
    - Agile and waterfall hybrid project management for iterative development.
- Engineering Standards:
    - **IEEE 1149.1:** Standard for test logic in integrated circuits, ensuring valid interconnections between ICs. [2]
    - **IEEE 1481:** Standard for analyzing chip metrics like timing and power consumption during IC design. [3]
    - **IEEE 1621:** Standard for user interface elements in power control of electronic devices. [4]
    - **IEEE 1680:** Standard for environmental assessment, focusing on sustainable material use and energy efficiency. [5]

Summary of Requirements

- Our system must accurately replicate the functionality of the ABC using modern technology.
- Our system must allow users to input data, view each step of the computation, and receive output.
- The system should use modern components.
- The computer must be a physical, tangible object that users can interact with.
- The computer design should fit within a workspace suitable for educational demonstrations.
- The user interface must be simple and intuitive.
- It must look cool.

Applicable Courses from Iowa State University Curriculum

- CPRE 281: Digital Logic
- CPRE 381: Computer Organization and Assembly Level Programming
- COMS 309: Software Development Practices
- COMS 227: Object-Oriented Programming
- COMS 228: Introduction to Data Structures
- EE 201: Electrical Circuits
- EE 230: Circuits and Systems in Electronics

New Skills/Knowledge acquired that was not taught in courses

- Arduino Coding
- EEPROM Programming
- Vacuum Tube Logic

Table of Contents

List of Figures

1. Introduction

1.1. Problem Statement

Our project is to recreate the ABC computer from scratch using modern technology. Our goal is to use the same logic that John Atanasoff used when developing the original ABC. We have designed our computer purely based on the resources that exist on the ABC. Most resources are written, there is a video from the 90s ABC reconstruction project, and we also used primary sources from people who worked on the ABC reconstruction project in the 90s. The purpose of our ABC recreation is educational, but also recreational. Most users will be students and professors in the pursuit of education, but also, computer hobbyists may enjoy using our computer or learning about it. Our project may benefit our society by creating more capable computer engineers. Our project is open-source, so globally, our project can benefit other schools or hobbyists by providing detailed documentation on how our ABC recreation works. Further educating computer engineering students through our project is important because visually seeing a computer's inner workings could make a difference in a student's success or even interest. Our computer is capable of education because it will have physical displays or LEDs that show what our computer is doing at that moment in time. Overall, our project could have a global impact on the education of students in a computer engineering-related field.

1.2. Intended Users

Computer Engineering Professor

1. The user is a professor in computer engineering, likely with in-depth expertise in computational systems and logic. They are responsible for teaching complex subjects, such as the architecture and function of computers, to students who may be new to these concepts. They need to convey these topics in a simplified and engaging way to help students understand.
2. Computer engineering professors need a way to demonstrate the function of a simple, easy-to-understand computer to their students because it helps them explain core computational concepts effectively.
3. The recreated ABC will serve as a tangible teaching tool, enabling professors to visually demonstrate how computers perform calculations and execute operations. This hands-on experience will make abstract concepts more concrete for students, enhancing understanding. The value of this product is in its ability to simplify complex ideas.

Computer Engineering Student

1. These users are students enrolled in computer engineering courses, often motivated to understand how computers work at a foundational level. They may have varying degrees of familiarity with computational logic and systems, but are driven by their interest in technology and their desire to succeed academically.
2. Computer engineering students need to learn about computational logic to build foundational knowledge in their courses and to succeed in the academic environment.

3. Students will benefit from interacting with the recreated ABC because it provides a real-world application of the theoretical knowledge they are learning in their courses. By witnessing how a computer performs calculations, students will deepen their understanding of computational logic, helping their academic progress.

Computer Hobbyist

1. This group consists of individuals with a personal interest in the history and development of computer technology. They may not be formally involved in academia or the computer industry, but they enjoy learning about how computers evolved and the first computers in the field.

2. Computer hobbyists need to learn about the ABC because they are interested in the history of computing technology.

3. hobbyists will find value in the project as it provides them with a tangible piece of computing history that they can explore and understand. By seeing a modern recreation of the first electronic digital computer, hobbyists can gain a deeper appreciation for the technological advancements that have shaped modern computing. This ties back to the project's mission to educate and inspire through historical recreation.

2. Requirements, Constraints, And Standards

2.1. Requirements & Constraints

Functional Requirements

1. The system must accurately replicate the functionality of the original Atanasoff-Berry Computer (ABC) using modern technology.
2. The system should be able to solve different systems of linear equations as the ABC did, using a similar computational logic.
3. The recreated computer must allow users to input data, view each step, and receive the correct solution as output.

Resource Requirements

1. The system should use modern components such as integrated circuits and semiconductor memory devices to emulate the original ABC.
2. Components must be sourced to align with IEEE 1149.1 and IEEE 1481 standards for testing and efficiency (constraint). [2][3]

Physical Requirements

1. The computer must be a physical, tangible object that users can interact with directly.
2. The design should fit within a workspace suitable for educational demonstrations, not exceeding typical desktop dimensions.
3. The system should visually display each step of the calculation process to enhance user understanding.
4. It must look cool.

UI Requirements

1. The user interface must be simple and intuitive, with clear instructions for students and hobbyists who may not be familiar with this type of background.

2. The interface should include visuals on how computations are completed.

2.2. Engineering Standards

Engineering standards are important because they ensure safety, reliability, and quality in everyday products and processes. They provide a common framework for all parts of the manufacturing process and guarantee collaboration across industries. Engineering standards help reduce errors and costs and ultimately enhance consumer trust. It is important to stay in line with engineering standards because they improve efficiency and help engineers maintain consistency in their work.

IEEE 754:

The IEEE 754 standard specifies methods for performing binary and floating-point arithmetic. It also specifies exception conditions and how they should be handled by default. This standard is intended to provide a common method for computation with binary and floating-point numbers that will yield the same result. This standard applies to computations being done directly through hardware, or through software. [6]

IEEE 1149.1:

The IEEE 1149.1 standard defines test logic for integrated circuits that standardizes approaches to multiple forms of testing. It provides a standard for testing interconnections between integrated circuits (ICs) that have been connected on a breadboard. It also provides a standard for testing the IC itself. This standard is intended to standardize the methods of testing ICs. [2]

IEEE 1481:

The IEEE 1481 standard defines methods for IC designers to analyze chip metrics. These metrics include timing and power consumption. Methods by which IC designers can express these metrics are also defined. This standard is intended to allow IC designers to more accurately and completely analyze semiconductor designs while expressing them in a way that is understandable to other IC designers. [3]

We believe IEEE 1149.1 and IEEE 1481 to have relevance to our project, but not IEEE 754. We believe IEEE 1149.1 and 1481 will have relevance to our project due to our use of ICs. We need a way to ensure that interconnections between ICs on our breadboards are valid, and we need to use the timing and power information given to us by the IC designers when considering our design. IEEE 754 is not so relevant to our project because it defines the modern way of performing binary arithmetic. Our project intends to use the same method of binary computation that was used on the original ABC, which does not follow IEEE 754.

IEEE 1621:

Standard for User Interface Element in Power Control of Electronic Devices. This standard guides the user interface designs' power control. Since we want our modified ABC to turn on/off or manage energy use, following this standard helps usability and consistency with other modern electronic devices. [4]

IEEE 1680:

Standard for Environmental Assessment of Electronic Products. This standard is friendly as it focuses on the environmental aspects of electronic products. Like recycling, energy efficiency, and environmentally friendly materials. This standard helps our project be more sustainable. [5]

3. Project Plan

3.1 Project Management/Tracking Procedures

Our project management style that we have adopted is a hybrid of the waterfall and agile approach. Our project benefits from this approach because it allows us to work on different components at the same time (agile), but within those components, the process is fairly linear (waterfall).

Our team tracks progress through the use of Discord and Github. Firstly, we use Discord as our primary method of communication where we let each other know what we have completed and what needs to be done next. Secondly, Github is used as our primary repository of files pertaining to the project, such as documentation and KiCad files.

3.2 Task Decomposition

1. Research Modules
    - 1.1. Add-Subtract Mechanism
        - 1.1.1. Look into modern approaches to replicate the ABC's adder-subtractor design.
    - 1.2. I/O Methods
        - 1.2.1. Study user interaction and techniques that keep the spirit of the ABC.
    - 1.3. Base Converters
        - 1.3.1. Analyze methods for data conversion to align with the ABC's requirements.
    - 1.4. Drum Memory
        - 1.4.1. Find out how the capacitor drums were used to store bits.
    - 1.5. Control Unit
        - 1.5.1. Design a modern equivalent of the control unit to manage data flow and operational sequences.
2. Design Modules
    - 2.1. Add-Subtract Mechanism
        - 2.1.1. Build adder-subtractor circuit in modern terms.
    - 2.2. I/O Methods
        - 2.2.1. Create a design that supports input and output for user interactions.
    - 2.3. Base Converters
        - 2.3.1. Finalize the base converter design to accurately handle data transformations.
    - 2.4. Drum Memory
        - 2.4.1. Design the memory in such a way that aligns with the original's spinning drum memory.
    - 2.5. Control Unit
        - 2.5.1. Define the unit's layout and operational logic to work with all modules.
3. Build Prototypes / Test Designs
    - 3.1. Add-Subtract Mechanism
        - 3.1.1. Assemble and test the add-subtract functionality on a breadboard.
    - 3.2. I/O Methods
        - 3.2.1. Prototype the I/O system and evaluate user interaction and functionality.
    - 3.3. Base Converters
        - 3.3.1. Test the base converters' accuracy and performance.
    - 3.4. Drum Memory
        - 3.4.1. Verify the functionality of the memory.
    - 3.5. Control Unit

            3.5.1. Prototype the control unit to ensure it successfully manages operations and data flow.

4. Build Final Breadboards
    4.1. Add-Subtract Mechanism
    4.2. I/O Methods
    4.3. Base Converters
    4.4. Drum Memory
    4.5. Control Unit
    4.6. Assembly of Breadboards
        4.6.1. Integrate all modules, creating a complete system.
5. Release Under Open-Source License
    5.1. Document Progress
        5.1.1. Comprehensive project documentation.
    5.2. Release to Public
        5.2.1. Allows educational and recreational use.
6. Final Deliverables
    6.1. ABC Finished
        6.1.1. All components are complete and meet functional and quality standards.
    6.2. Released to the Public Under an Open-Source License

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

1. Research Modules
    - Milestone: Completion of research for all core modules.
    - Metric: Documentation of research findings for each module.
    - Evaluation: Achieved if the documentation provides clear guidance for all phases.
2. Design Modules
    - Milestone 1: Finalize design choices for each module.
        - Metric: Completion of design documents for all modules.
        - Evaluation: Consensus by the team for all design choices.
    - Milestone 2: Review and revise designs based on feedback.
        - Metric: Documented revisions and final approach for each design.
        - Evaluation: Each Consensus by the team for all design choices.
3. Build Prototypes / Test Designs
    - Milestone 1: Breadboard prototypes assembled for each module.
        - Metric: Prototypes assembled and tested for functionality.
        - Evaluation: Prototypes meet 95% success rate in functional tests with adjustments.
    - Milestone 2: Complete prototype testing for all modules.
        - Metric: Pass rate of at least 99% in functionality tests for each module.
        - Evaluation: Prototype achieves expected results.
4. Build Final Breadboards
    - Milestone 1: Assemble breadboards with final design for each module.
        - Metric: Each circuit meets design specifications and functional tests.
        - Evaluation: Breadboard circuits pass functionality, compliance, and safety tests, with adjustments documented as necessary.
    - Milestone 2: Integrate modules into final breadboard assembly.
        - Metric: Fully integrated design meets performance and interconnection standards.
        - Evaluation: Final integration passes compliance tests (IEEE 1149.1 for testing interconnections), meeting all technical and functional requirements. [2]

5. Release Under Open-Source License
    - Milestone 1: Complete project documentation for public release.
        - Metric: Documentation is peer-reviewed and approved for clarity and completeness.
        - Evaluation: Documentation meets open-source standards, making the project accessible and understandable for external users.
6. Final Deliverables
    - Milestone: Final quality check and project sign-off.
        - Metric: Final system meets all project specifications and passes final user and performance tests.
        - Evaluation: Project receives final approval for completion and is available under open-source, fulfilling educational and functional objectives.

3.4 Project Timeline/Schedule

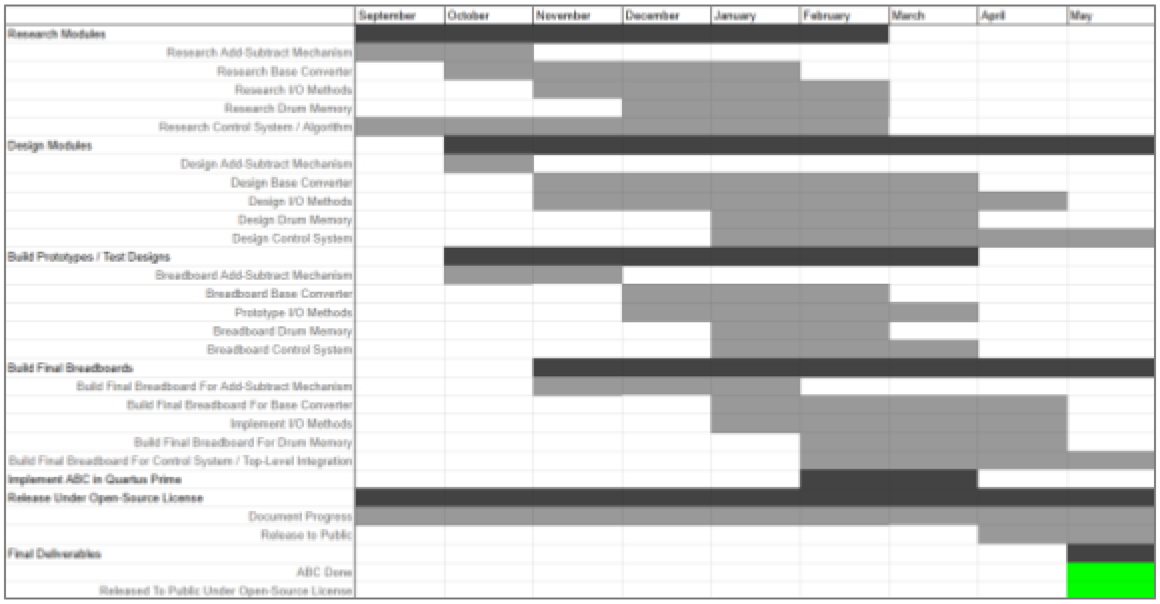


Figure 1: Gantt Chart

Our project's tasks will be performed in the following order: Research Modules, Design Modules, Build and Test Modules, Build Final Breadboards, and Release Under Open-Source License. We researched the different modules from September-February. We designed the modules one by one after gathering enough information from October-April. We built prototypes and tested our designs directly after finishing said designs, this phase spans from October to March. We built all of our modules and our high-level module on breadboards after testing our designs, this happened from November to May. We also documented our progress throughout the entire project, and we will release it to the public in May. The final deliverables of our project are the working ABC reconstruction, and that it is released under an open-source license. The ABC reconstruction was finished in May, and we will soon release it to the public under an open-source license.

3.5 Risks and Risk Management/Mitigation

Task	Risks	Mitigation Plan
Research Modules	There is not enough information for us to 100% faithfully recreate a module. Risk factor: 0.9	If there is a similar component from that time period, we will use it for inspiration and make our own version of the module, otherwise, we will use modern techniques to achieve the same goal as the original.
Design Modules	Making a mistake while designing. Risk factor: 0.95	We have tools that are testing our designs. If we design something incorrectly, we can test it, and then fix our error.
	We run out of time in the school year to design all modules of the ABC. Risk factor: 0.05	
Build Prototypes / Test Designs	Making a mistake while building a prototype. Risk factor: 0.8	We will have debugging outputs in our prototypes when constructing them. If we make a mistake while building a prototype, we will be able to tell what is wrong, and we will fix the error.
	We run out of time in the school year to build prototypes of each module. Risk factor: 0.1	
Build Final Breadboards	There is a malfunction in the breadboard. Risk factor: 0.2	
	Our final design is incorrect and does not function on the final breadboard. Risk factor: 0.3	
	We run out of time in the school year to build all final breadboards.. Risk factor: 0.15	Start working harder. We will provide enough documentation for a future team to create a complete implementation.

Release Under Open-Source License	We do not follow the guidelines for releasing our project under an open-source license. Risk factor: 0.1	

Figure 2: Risk Management Chart

We ran into a few of these risks during our project. Information on the ABC was extremely hard to come by, and as a result, we ended up making some of our own designs on modules that we could not 100% faithfully recreate. One such example is when we were designing the base-10 punch card interface. We were unsure how exactly it was able to tell when a number was negative and we decided to make an extra column of our punch card distinguish that fact. There were times where our prototypes were non-functional. This was always remedied by looking over our circuit and fixing any mistakes we made. Also, we risked running out of time to complete our final breadboard implementation. As a result, we upped our man-hours and worked extra hard to complete it.

3.6 Personnel Effort Requirements

Task	Details	Estimated Hours	Actual Hours
Research Modules	Research the functionality and implementation of the modules	160 Hours	240 Hours
Design Modules	Design modules of the ABC based on our research and understanding of how each module should function.	200 Hours	200 Hours
Build Prototypes / Test Designs	Build prototypes of components on breadboards, and test components individually.	300 Hours	200 Hours
Build Final Breadboard Implementation	Build final implementation of the ABC on breadboards.	250 Hours	350 Hours
Release Under Open-Source License	Document the Process and Release our Designs	50 Hours	50 Hours

Figure 3: Personnel Effort Requirements Chart

Originally, we believed researching the ABC would be easier. We learned that there aren't many sources covering the intricacies of the machine. As a result of this, we ended up spending a lot more time researching than we thought we would. Building the prototypes did not take as long as we originally thought. We were slowly running out of time to build our computer and decided to hold back on the amount of prototyping we did. A few designs ended up going straight to our final implementation. And finally, building the final implementation took longer than we thought it would. It took a lot of effort and time to decide how to place all of the breadboards, and assemble them all together. A lot of testing on the final implementation took place.

3.7 Other Resource Requirements

This project involves a myriad of different parts, materials, and other pieces that will be needed to eventually realize all of the requirements and expectations. Since this project is very heavy on the circuit design component, we will be using and recording an extensive parts library that will include all integrated circuit chips, light-emitting diodes, resistors, capacitors, headers, and connectors we will use throughout this project. Along with all those components, we will also make extensive use of wiring materials and breadboards. Finally, our project contains a significant amount of lab time that will require our attention to manage along with all the other facets to bring about our successful completion of this project and this course.

4. Design

4.1 Design Context

4.1.1 Broader Context

Our design problem exists in the context of education and historical preservation. Implementing the Atanasoff-Berry Computer with modern technology will help connect students with the roots of modern computing by allowing professors to use our design when teaching students about the history of computers and digital logic. It will also benefit those who are simply interested in the history of computing, as our design will be open-source and allow them to take a deep dive into the innermost workings of our design.

Our project will affect the general well-being of students, professors, and those interested in the history of computing by indirectly improving their intellectual welfare by fostering curiosity about technology and engaging them in learning. Professors will be directly affected because they will gain an educational tool. Students will be able to gain knowledge using a physical example designed to help them learn. Those who are interested in the history of computers will also be affected because they will have the ability to learn about the first electronic digital computer like never before.

Our project reflects the values, practices, and aims of the cultural groups it affects positively. It furthers the education abilities of professors and allows students a chance to physically learn about computing. It also enhances the computer hobbyist community's understanding of the first electronic digital computer.

Our ABC implementation project may have a negative impact on the environment. We are unsure of the practices of the manufacturers of our components, and we may be indirectly harming the environment by using their services. Our project will also increase the energy usage from nonrenewable sources because it has to consume power to run.

Our project will not have any funding issues due to the somewhat low cost of components. We plan to use two Android tablets in our design, which will consume the bulk of our budget. However, we will target Android devices that are just powerful enough for our simple apps in order to save on cost for us, and others who would like to build our ABC implementation on their own.

4.1.2 Prior Work/Solutions

There have been two projects in the past here at Iowa State University relating to the ABC. The two projects are the original construction of the ABC by John Atanasoff and Clifford Berry, and the reconstruction performed by a team of engineers led by John Gustafson in the 90s. The goal of the reconstruction project was to create a 100% faithful reconstruction of the original ABC to prove that it was a functional machine. The reconstruction team completed their goal in 1997.

	Our ABC Reconstruction Project	90s Reconstruction Project
Pros:	• Smaller scale • We are did it all electronically • Easier to debug with modern tools • Lower cost • Faster development cycle • Ability to use modern tools like computer simulations	• Closer to the time period of the original ABC construction • Captures the spirit of the era in physical form • Greater historical accuracy in design and components
Cons:	• Fewer people • Lack of historical authenticity in hardware • May not fully represent the constraints of the original project	• Requires sourcing obsolete components • Increased risk of mechanical and electrical failures • Longer project timeline due to complexity

Figure 4: Prior Works Compared to Our Project Pros/Cons Table

The following references outline the previous works that have been completed pertaining to the ABC:

[1] A. Burks and A. W. Burks, "The First Electronic Computer: The Atanasoff Story." Ann Arbor: University of Michigan Press, 1988.

[2] J. Gustafson and C. Shorb, "The Atanasoff-Berry Computer In Operation," YouTube, https://www.youtube.com/watch?v=YyxGIbtMS9E&ab_channel=ComputerHistoryMuseum (accessed Dec. 7, 2024).

[3] J. Gustafson, Reconstruction of the Atanasoff-Berry Computer, http://www.johngustafson.net/pubs/pub57/ABCPaper.htm (accessed 2024).

4.1.3 Technical Complexity

Our design consists of multiple subsystems, such as the add-subtract mechanism, base converter, memory drums, and our I/O solution. Each subsystem leverages distinct scientific, mathematical, and engineering principles:

- **Add-Subtract Mechanism:** Implements binary arithmetic through modern logic gates replicating the logic of the original ABC.
- **Base Converter:** Utilizes modern digital logic circuits and ICs to replicate the function of the original ABC's base converter circuit.
- **Memory Drums:** Simulates temporary data storage using modern memory technologies.
- **I/O:** Uses Android app interfaces for data input and output.

Our project includes challenging requirements that match industry standards. Our project must replicate the ABC's functionality using modern components while maintaining the spirit of the original design. This requires not only technical ingenuity but also historical research and adaptation. The use of modern ICs and Android apps introduces challenges in achieving functional fidelity while adhering to industry standards. Utilizing the modified Gaussian elimination process that the original ABC used further increases the complexity of math and algorithms used in our project.

4.2 Design Exploration

4.2.1 Design Decisions

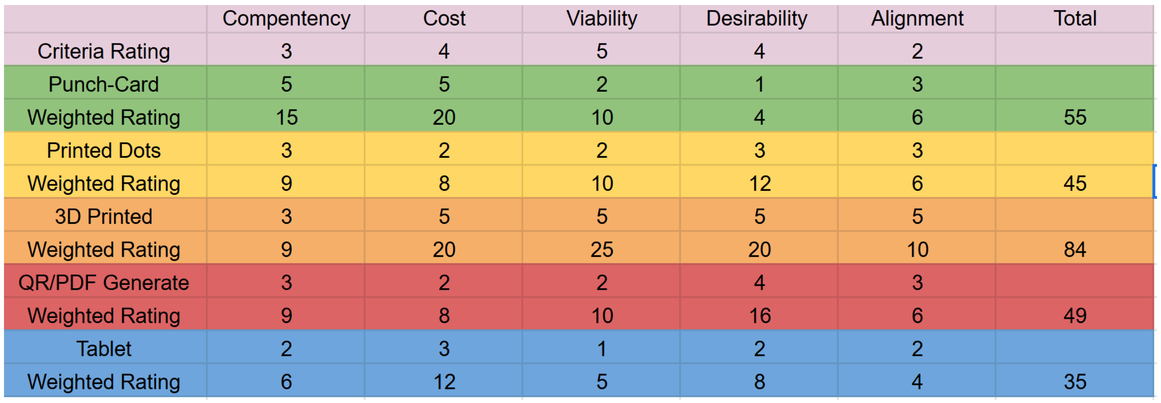
1. Input / Output
    a. Input / Output corresponds to how we will receive the equations to compute from the user (the original used physical punch cards), and how we will output the results to the user (the original used a rotary dial mechanism.
    b. We decided to use Android apps for base-10 input and base-2 input/output, and we decided to use 7-segment displays for base-10 output. We made these decisions because we do not have the resources to develop our own physical input/output methods as the original did. These decisions have helped us to complete our design. Re-creating the original's input/output methods 100% faithfully would be an entire project itself.
2. User Feedback
    a. User Feedback corresponds to how we will show the computer's calculations to the user.
    b. We decided to use LEDs to visualize the function of the machine. This will allow others to understand how the machine works given that they have used our resources to learn what the goal of each module is. This decision has helped us to speed up our design by allowing us to see what the machine is doing.
3. Physical Layout
    a. The physical layout corresponds to how we will be assembling our final design.

    b. We decided to use breadboards and tablets attached to a large piece of plywood as our final assembly. We originally planned on using PCBs in our final design, however, towards the beginning of the second semester, we realized this goal was unrealistic. This decision has helped us to develop a more complete version of our recreation.

4.2.2 Ideation

Our project allows for open-ended conversations about design choices because we are implementing modern technology with the Atanasoff-Berry Computer. Our most talked about component in the project is how we will handle input and keep the user involved. The original input for the computer was punch card-based, which was the first option we discussed. We want to stay loyal to the original machine, so we sought out IBM punch card machines but struggled to find any. Our second consideration was printing dots on a piece of a standard sheet of paper and scanning the paper as the input. Relating to the previous consideration, our third consideration, and a little bit far-fetched, was 3d printing a punch card based on the user's input. Our fourth idea for the component was to generate a QR code or a PDF that our machine could read. Our fifth and final option is using tablets as input. We arrived at this idea because it is feasible for our team to program a punch card app based on the original machine. This way, we could also stick with our main idea of visualizing the machine's operation.

4.2.3 Decision-Making and Trade-Off



	Compentency	Cost	Viability	Desirability	Alignment	Total
Criteria Rating	3	4	5	4	2	
Punch-Card	5	5	2	1	3	
Weighted Rating	15	20	10	4	6	55
Printed Dots	3	2	2	3	3	
Weighted Rating	9	8	10	12	6	45
3D Printed	3	5	5	5	5	
Weighted Rating	9	20	25	20	10	84
QR/PDF Generate	3	2	2	4	3	
Weighted Rating	9	8	10	16	6	49
Tablet	2	3	1	2	2	
Weighted Rating	6	12	5	8	4	35

Figure 5: Weighted Decision Matrix

The above matrix shows how we came to the conclusion of using tablets for punch cards. Competency refers to how it relates to our team's skill set. Cost refers to the cost required to adopt the new idea. Viability determines if the idea is applicable in real life. Desirability refers to how the user accepts and interacts with the new idea. Alignment refers to how the idea aligns with our project strategy. A lower total score is in our better interest and tablets align with that score.

4.3 Proposed Design

4.3.1 Overview

At its core, our design strives to strike an adequate balance between remaining faithful to the original ABC in spirit and function while also embracing certain aspects of modern computer design developed within the most recent half century. While a more modern design may prove easier to implement with modern technology and more understandable by those familiar with modern components, a design too modern in its approach loses all historical connection to the original machine, and thus the university itself. Therefore, we have opted to make the following design decisions in regards to the original machine's design as follows:

1. Modified Gaussian Algorithm - Since the machine was designed to solve systems of linear equations with only addition and subtraction operations, the original device built by Dr. Atanasoff and Clifford Berry used a modified version of the standard Gaussian elimination process for solving linear systems. In our design, we have opted to make use of this same algorithm as well, limiting our machine to only using addition and subtraction in the process.
2. Serialized Computation - The original ABC, with its data stored on rotating drums of capacitors, performed all of its computations in a serialized fashion starting with the least-significant bit of each number. This is markedly different from how modern computers function, however, we have decided to preserve this characteristic of the old machine for that very purpose as it maintains the novelty of our design when compared to modern systems.
3. Excess Human Interaction - With modern computers, there exist many different forms of automation that carry out different tasks or computations without human involvement, and while the original ABC had some autonomous actions itself, the steps for using the ABC to solve systems of equations still involved a great deal of human interaction due to its limited technological capabilities. Therefore, we have opted to retain many of these such

interactions, as opposed to automating them away, as we believe they are vital to the overall spirit of using the machine as its predecessor was also used.

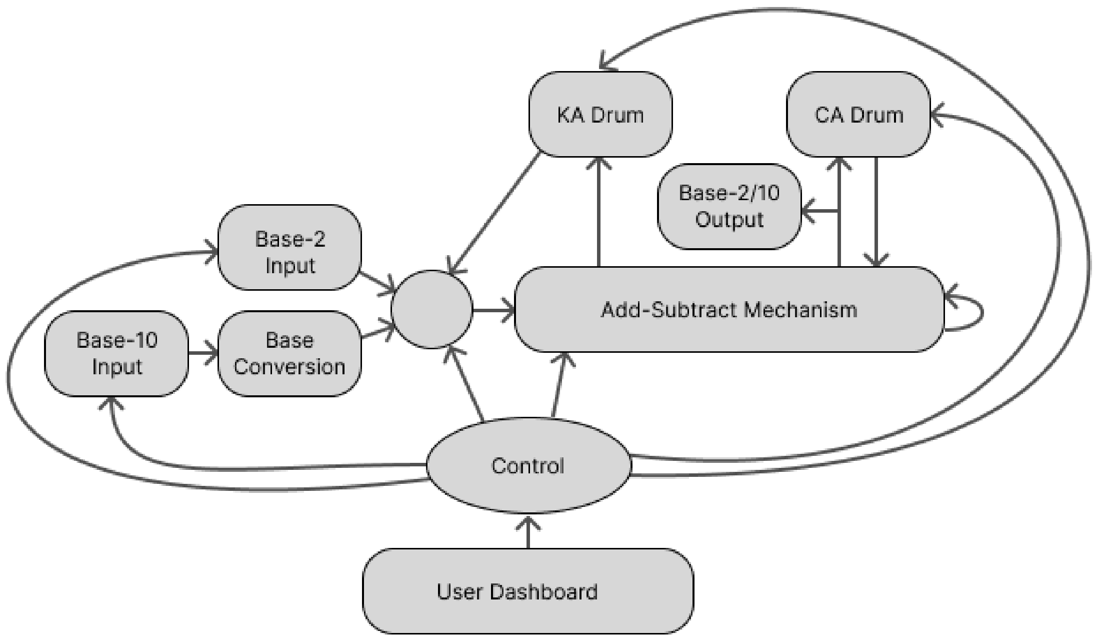


Figure 6: Our ABC Design Block Diagram

With respect to these three fundamental design decisions to preserve our historical connection to the ABC, the rest of our design consisted of more modern aspects in somewhat varying degrees. Figure 6 contains a block diagram of all the core components of the original machine that we sought to implement with modern technology. Each of these components were developed first by analyzing how the component functioned in the original machine through extensive research, then creating a modern equivalent via digital logic circuits and verilog code in Quartus Prime for simulation, and finally implemented with real devices in a few key ways. Most commonly, components such as the KA and CA drums, the Add-Subtract Mechanism, and the Control were done via Transistor-Transistor Logic (TTL) integrated circuits on breadboards, while the data input modules such as the Base-10 Input and Base-2 Input were realized with the help of some custom Android software in communication with a handful of ESP32 microcontroller boards.

4.3.2 Detailed Design and Visual

To highlight our design and both its similarities and differences to the original ABC, this section will focus on diving into the details of each component, how it functioned in the original machine, and how it was simulated using digital logic circuits in Quartus Prime.

Timing Drum

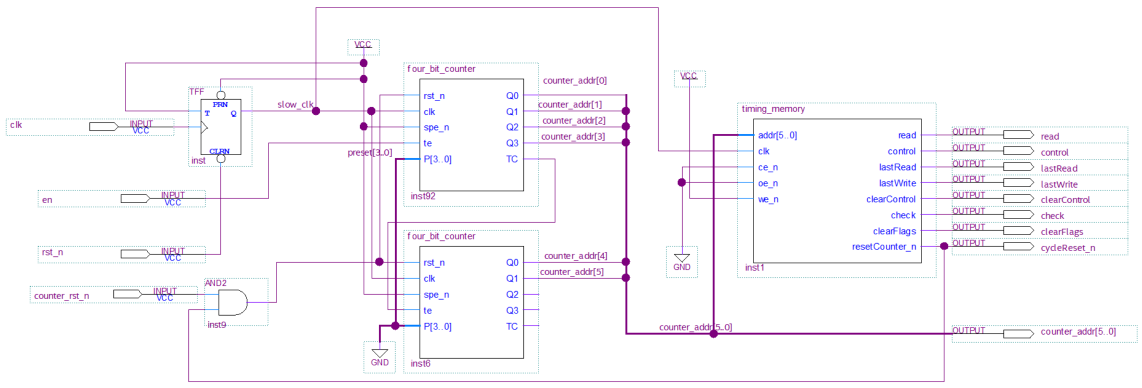


Figure 7: Timing Drum Digital Circuit

Core to the ABC's hybrid mechanical and electrical design was a set of four rotating drums that all spun on the same axle. The two largest of these drums were the KA and CA drums, and these drums stored the coefficients of two linear equations, one on each drum, for computation as part of the larger system. Rotating with these two, there existed a decimal to binary conversion drum that served to convert decimal inputs into the machine into their binary equivalents, and finally, a timing drum that provided a series of timing signals at different points in the rotation shared by all of these drums. These drums, minus the timing drum, rotated through 60 different 6° positions along the drum's surface with the first 50 such positions being populated with capacitors for data storage while the last 10 remained blank to allow the machine time to switch control signals.

To replicate this rotational motion of the drums in a modern sense while also providing the same signals as the timing drum, we designed the circuit in Figure 7 as a general timing drum that consists of two four-bit binary counters alongside a memory chip. These counters count through the 60 different positions of the drums and feed those bits as an address into an EEPROM that outputs pre-programed signals at each necessary position. Here it is important to note that this module also contains a flip-flop to divide the clock in half and only count every other clock cycle. This will be explained later in the next section.

Memory Drums and ASM Muxes

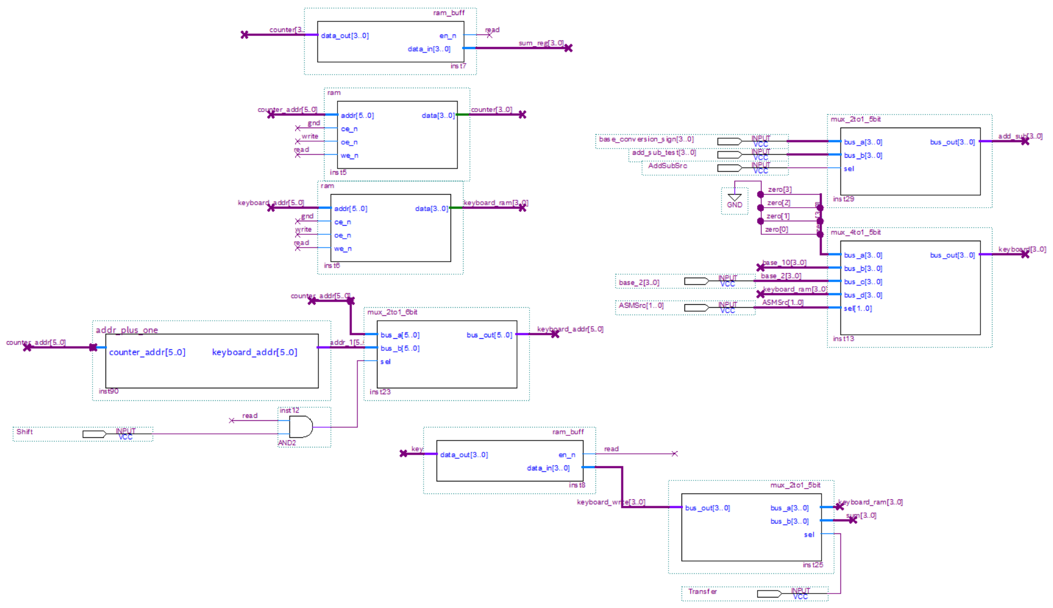


Figure 8: Memory Drums and ASM Muxes Digital Circuits

As explained previously, the drums featured 50 bit positions that were read from sequentially, and by making use of the drum's rotation, these bit positions were also able to be written to at the same time another was read, provided the writing occurred one position later in the drum's rotation. This formed the core computational loop for the original machine as the data would be read from the memory drums, passed through the Adder-Subtractor Modules (ASMs), and then written back to the CA drums specifically to store the result.

The circuit above implements the same idea by using two SRAM chips to serve as the memory drums. These chips take a six-bit address from the timing module that represents the rotational position of the drum, and then is selectively read from and written to at each address on two different clock cycles. While not a perfect representation of the simultaneous reading and writing done by the original machine, we believe that this process can approximate its behavior as best as possible given the modern tools available. The data from these RAM chips are then passed through a few MUXes that determine the inputs into the ASMs, while the data for writing back comes through two I/O buffers to maintain data integrity. Finally, since the original design also features a shifting mechanism for the KA drum, that RAM's address can be slightly altered via a selection between the same address as the other memory drum, or the next address in the rotation to achieve a rightward shift.

Punch Card Input and Output

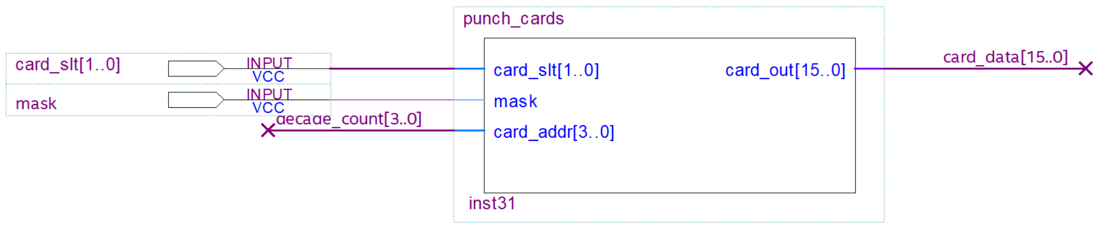


Figure 9: Punch Card Input and Output Circuit

As a product of the time period in which the original ABC was built, all of the machine's data entry was done via punched cards in one of two formats. Firstly, using the standard IBM punch cards of the time, a series of decimal numbers could be entered into the machine that were then converted into binary for computation. Secondly, Atanasoff and Berry also developed their own method of punching cards directly from the binary numbers that the machine was capable of both reading and writing.

In the Quartus simulations performed for our digital logic based design, these punch cards were merely simulated via a verilog script that would output the necessary data for both types of punch cards used. More information on the Android software and microcontroller implementations of these punch cards can be found in section 6.

Decimal to Binary Drum

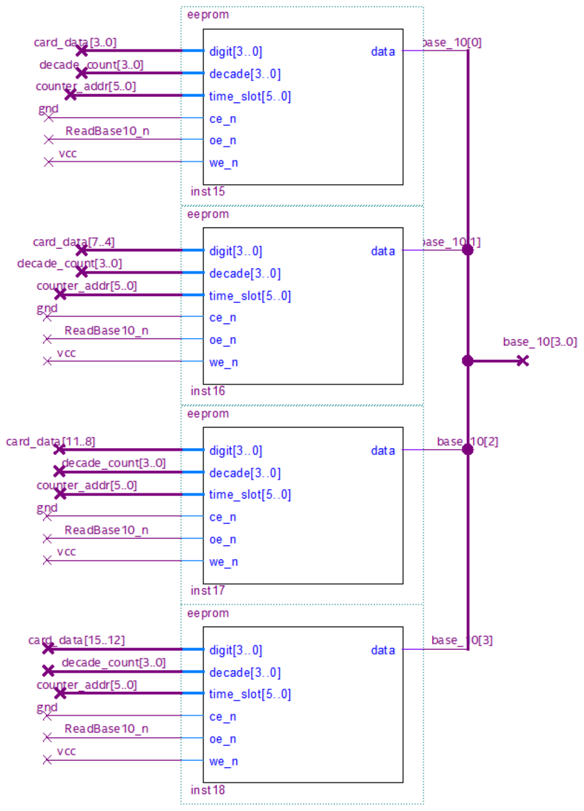


Figure 10: Base Conversion Circuit

As mentioned previously, one of the key drums contained within Dr. Atanasoff's device was a drum dedicated to converting decimal numbers into their binary equivalents. With the help of a punch card reader, this drum effectively contained a binary lookup table of all the digits from one to nine in their binary form, along with the binary forms of each of those digits up to the 15th decade (1, 10, 100, 1000, etc.). Reading a number from a base 10 punch card then consisted of first reading the most significant digit, finding the corresponding ring of pegs on the conversion drum, and then adding that number with zero and storing the result on the CA drum. Each subsequent conversion was then added to the previous results to get the full binary representation of the decimal number. Since each base 10 punch card could contain up to five coefficients, the drum was capable of converting all these numbers at the same time in a parallel fashion.

For our design, we chose to implement the binary lookup table with an EEPROM chip that could be pre-programed with the proper values across a determined encoding scheme for its address. The four most significant digits of the EEPROM's address were reserved for the specific decimal digit being converted, then the next four digits were used to represent the decade scaling factor as a power of 10, and finally the last six bits were set as the same rotational address used for the memory drums. Our design contains four of these EEPROMs to emulate the parallelism demonstrated by the original machine for this process, albeit at a slightly smaller scale.

Adder-Subtractor Mechanisms

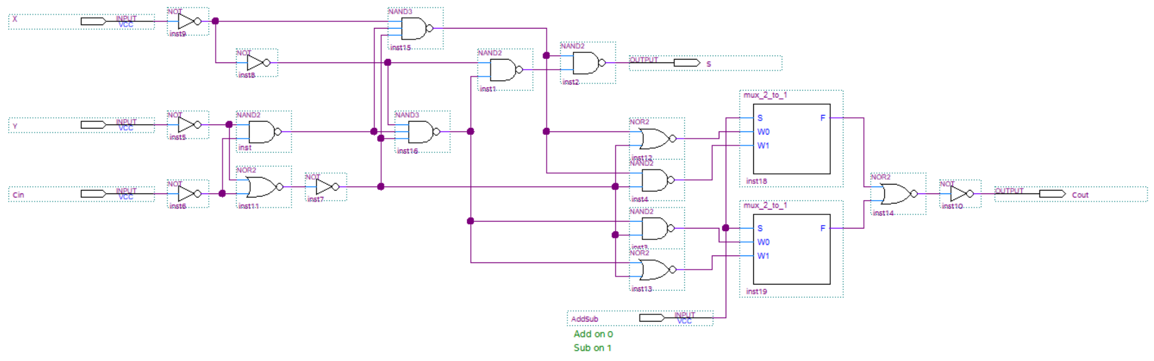


Figure 11: ASM Circuit

Consisting of seven dual triode vacuum tube devices, the ASMs within the original machine were responsible for both the addition and subtraction computations that the machine's algorithm required. While not particularly well versed in the intricacies of vacuum tube logic circuits, our research was successful in uncovering a recreation of these circuits using modern digital logic gates, which we were able to modify slightly to get the design above. It is important to note that both this circuit and the original one are capable of handling two's complement representations of negative numbers, which means that both designs are capable of performing computations with respect to signed numbers.

Carry Drum

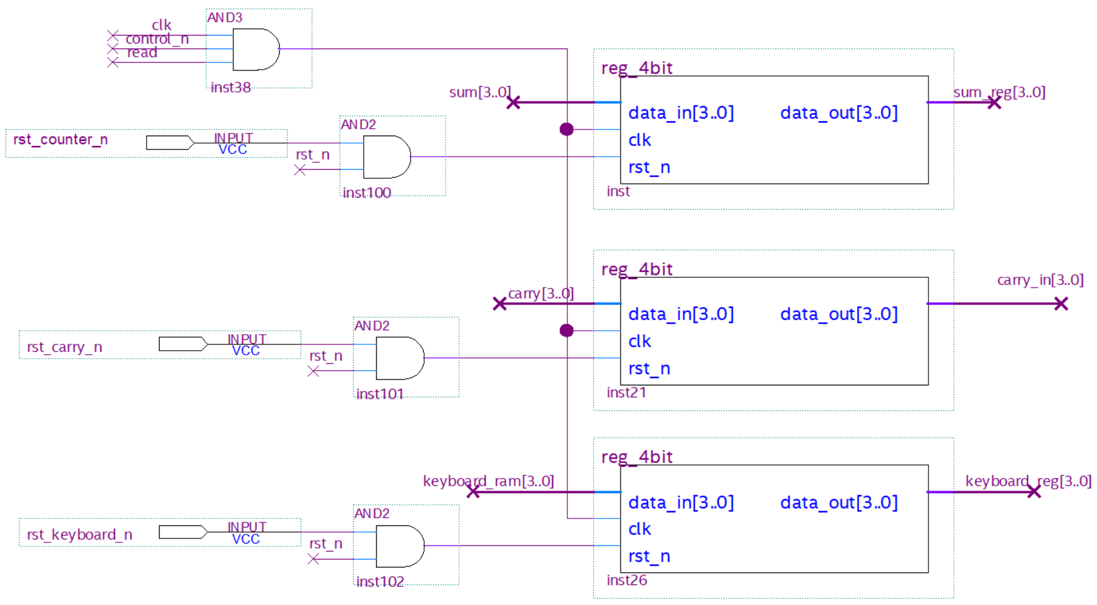


Figure 12: Carry Drum Circuit

While the main four drums of the ABC all shared the same axis of rotation, there existed one final drum within the machine that rotated separately from these four. This drum, which was coupled with yet rotated faster than the other drums, was used to store the carry of each subsequent computation for use in the next addition or subtraction operation. As mentioned before, the operations done within this machine were performed serially, meaning that each computation started with the least significant bit of two numbers and propagated on up through to the most significant bit of the two numbers, and if a carry were computed at any stage at or between, it would be stored on this drum for the next operation concerning the next bits in the two numbers.

To realize this within our design, we opted to make use of a register to store the carries for each computation which would be controlled by the same signals controlling the reading from and writing to the memory chips. In an effort to synchronize the timing of these carries with both the sums and the data from the KA drum, two additional registers were used despite them not being fully represented in the original machine. This deviation is again due to some non ideal properties of modern logic circuits that do not lend themselves well to this type of serialized design.

4.3.3 Functionality

Our design is intended to solve systems of linear algebra equations that are taken from the user's input. The equations that the user inputs will be translated onto a virtual punch card, and then be converted to binary. Once the equation is in its binary form, the user uses the ABC to solve for the unknown variables present in the equation by utilizing adders, memory drums, and base-2 punch card tablets to perform the Gaussian elimination algorithm. Once the equation has been solved, the user can output the results onto an array of 7-segment displays.

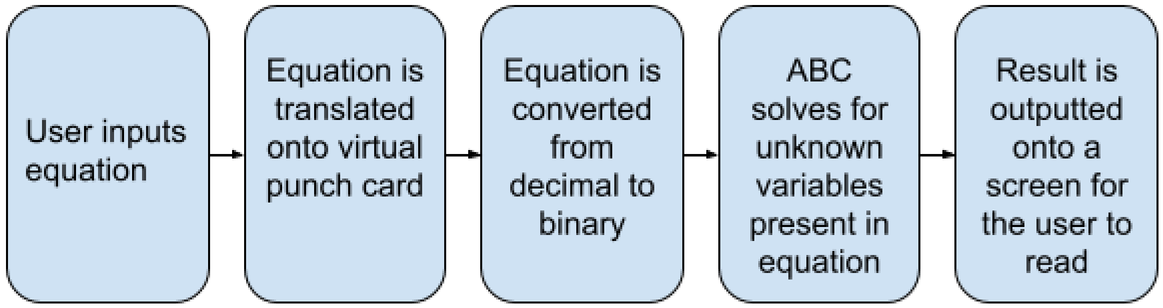

User inputs equation
Equation is translated onto virtual punch card
Equation is converted from decimal to binary
ABC solves for unknown variables present in equation
Result is outputted onto a screen for the user to read

Figure 13: Functional Flow of Our ABC Recreation

4.3.4 Areas of Challenge

We faced a significant amount of challenges in completing our design. Our biggest challenge was finding enough information on the machine to feel comfortable calling our design a faithful reconstruction. Another challenge we faced was actually designing our machine based on the information we could find. And another big challenge we faced was building our design on breadboards.

We overcame our challenge of finding enough information to feel comfortable calling our design a faithful reconstruction by doing copious amounts of research. We were eventually able to find a few key primary resources in legal documents, and design diagrams and descriptions from the library.

Actually designing our machine was a huge milestone in our project progress. Using the primary diagrams and the descriptions we did have made it challenging to come up with a design. The ABC works entirely in parallel, so building a full-fledged recreation would take way too many chips for our project. We decided to build a mini version that could handle equations with 4 coefficients.

We are currently facing the challenge of building our design on breadboards. Bugs are easy to come by and meticulously placing each wire takes a lot of time. We are actively mitigating this challenge by pouring tons of man-hours into this step. Overcoming all of these challenges was not easy, but it has helped us to get to a good spot in our implementation. No matter what, we will have completed something we are proud of.

4.4 Technology Considerations

1. Integrated Circuits (ICs)

Strengths: Integrated circuits offer a compact, reliable, and energy-efficient solution compared to the vacuum tubes used in the original ABC. ICs allow us to implement complex logic functions within a smaller physical footprint, making the design more feasible and scalable for modern educational use.

Weaknesses: Using ICs, while efficient, moves away from the original design's reliance on discrete vacuum tube circuits. This departure sacrifices some historical accuracy in exchange for practicality, as vacuum tubes are difficult to source and maintain.

Trade-Offs: We opted for ICs due to the reduced power requirements and simplified design process. Additionally, ICs enable us to achieve the desired computational logic without the extensive space and cooling needs of vacuum tubes, making the ABC replication more manageable and user-friendly.

2. Android Tablets

Strengths: Android tablets provide a versatile, interactive interface for both input and output, which simplifies user engagement and feedback. The use of tablets allows us to simulate the punch card system for input and visualize binary data transformations, which is crucial for educational clarity.

Weaknesses: While convenient, tablets are not a true reflection of the original punch card-based input system, which may detract from the historical authenticity of the replication. Furthermore, tablets require software development to emulate punch card functionality, adding complexity to the project.

Trade-Offs: Tablets were selected due to their accessibility, ease of programming, and ability to simulate various forms of input and output. This approach balances educational value with practical usability, allowing users to interact with the system while maintaining an approximation of the original input/output experience.

3. Breadboard and PCB Design

Strengths: Breadboards allow for easy prototyping and testing, enabling us to validate circuit designs before creating the final breadboard implementations.

Weaknesses: Breadboards are limited in terms of space and connection stability, which can complicate complex circuits.

Trade-offs: We are using breadboards for initial prototyping, allowing us to refine the circuit design. Once designs are finalized, we will implement them fully on breadboards. Using breadboards is faster and easier to debug then PCBs.

5. Testing

5.1 Unit Testing

The individual components of our design were each tested separately. The design of our components were tested early in the design stage using programs to simulate digital logic. After finishing the design, we built the components on a breadboard and tested the design physically by using LEDs and a multimeter to measure important voltages not represented by LEDs. After ensuring our design worked, we finalized the implementation on our interconnected breadboard system.

5.2 Interface Testing

Our design features many different interfaces that include many of the individual modules that make up the machine. While our unit testing ensured that each module works by itself, we combined certain modules together to cover key interfaces, like machine input/output, data conversion and storage, and finally, computation. Once we passed our unit testing, we then moved on to implementing each of these key interfaces by stringing together our modules and ensuring they function as expected. Android interfaces were tested using our ESP32s to ensure that correct data was being transferred between our machine and apps.

5.3 Integration Testing

From our requirements, we needed to ensure that the machine can perform all the same functions as the original ABC while taking input and output in an analogous manner. Thus, the primary integration path for us is to take in base ten input, convert it into base-2, store it in the machine's memory, perform one computational step, and then output the base-2 data for later use. The secondary integration path consists of many of the same steps, however, the input needs to come in as base-2, so no base conversion will be necessary. This testing had to wait until we completed our designs and started combining breadboard modules.

5.4 System Testing

All of our different stages of testing were necessary to verify the overall system performance and ensure that we met all of our requirements. Specifically, we needed to run unit tests for every module we created, interface tests on the three different key operations for our machine, and finally, some higher-level integration tests that make sure the machine can fully complete a computation step. Then we used those steps to prove that our design can fully solve a system of linear equations.

5.5 Regression Testing

We implemented some regression testing by making sure that any new hardware design worked properly according to our unit tests on breadboards first. Then, by making sure all designs were breadboarded first, we could ensure that they would not break any old functionality.

5.6 Acceptance Testing

We demonstrated our meeting of design requirements by making sure we tested as often and as thoroughly as we can within the presence of our advisor so that he too could verify our findings. By actively involving our advisor/client in the design and verification process, we hope to further deliver a useful educational aid to him and future students.

5.7 User Testing

With the current state of our implementation, we have been unable to complete user testing. If we were able to complete user testing, we would have given a demonstration of our machine and note down the users' reactions. Afterwards, we would have the user answer some questions to ensure they were able to understand how the machine worked and get any recommendations.

5.8 Results

The results of our testing were mostly successful. Our unit testing stage was very successful in ensuring our designs worked. The interface testing proved successful in that our Android apps worked exactly as planned. We are still working through the integration and system testing phases. At this point, our design is very nearly fully implemented on breadboards. We are slowly working through bugs in the system that we are finding due to our system testing. We have worked very closely with our advisor throughout all stages of our design, as a result, we have been performing acceptance testing during our entire project.

6. Implementation

Android Apps/ESP32s:

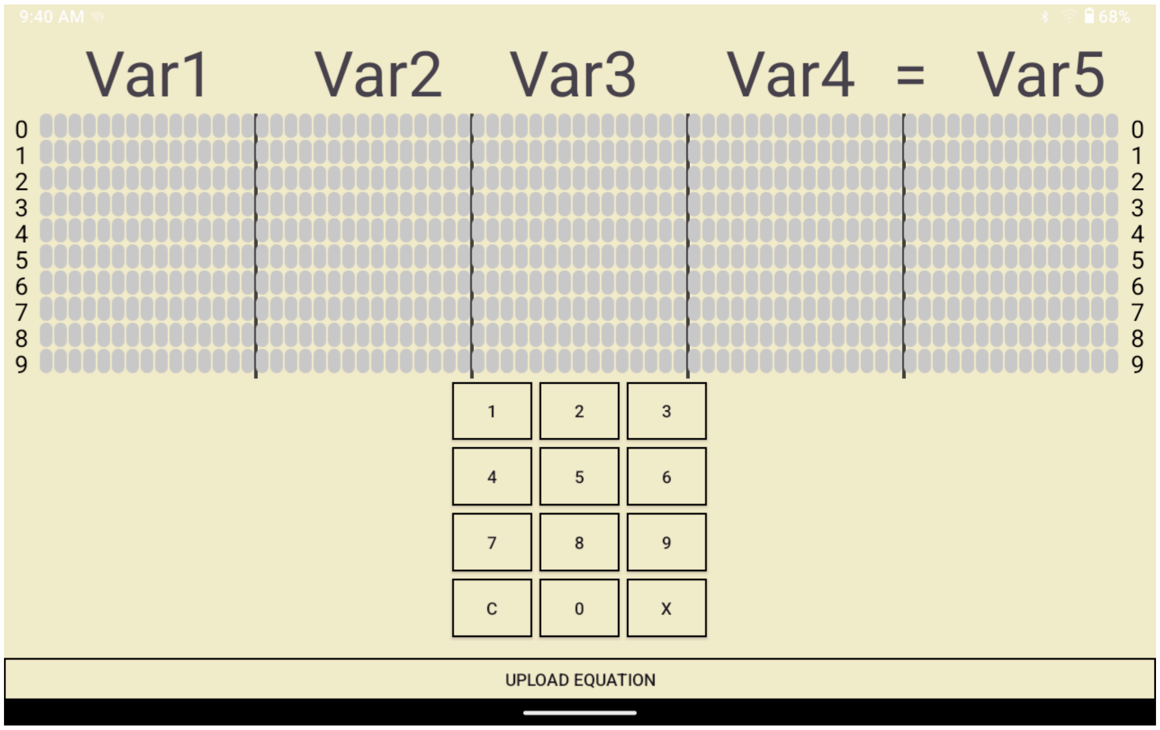


Figure 14: Android Base-10 Punch Card App

The Android base-10 punch card app is for inputting base-10 data into the machine. The data from the app is uploaded to an ESP32 that turns it into readable base-2 for our machine. Our system then takes this data and stores it into RAM.

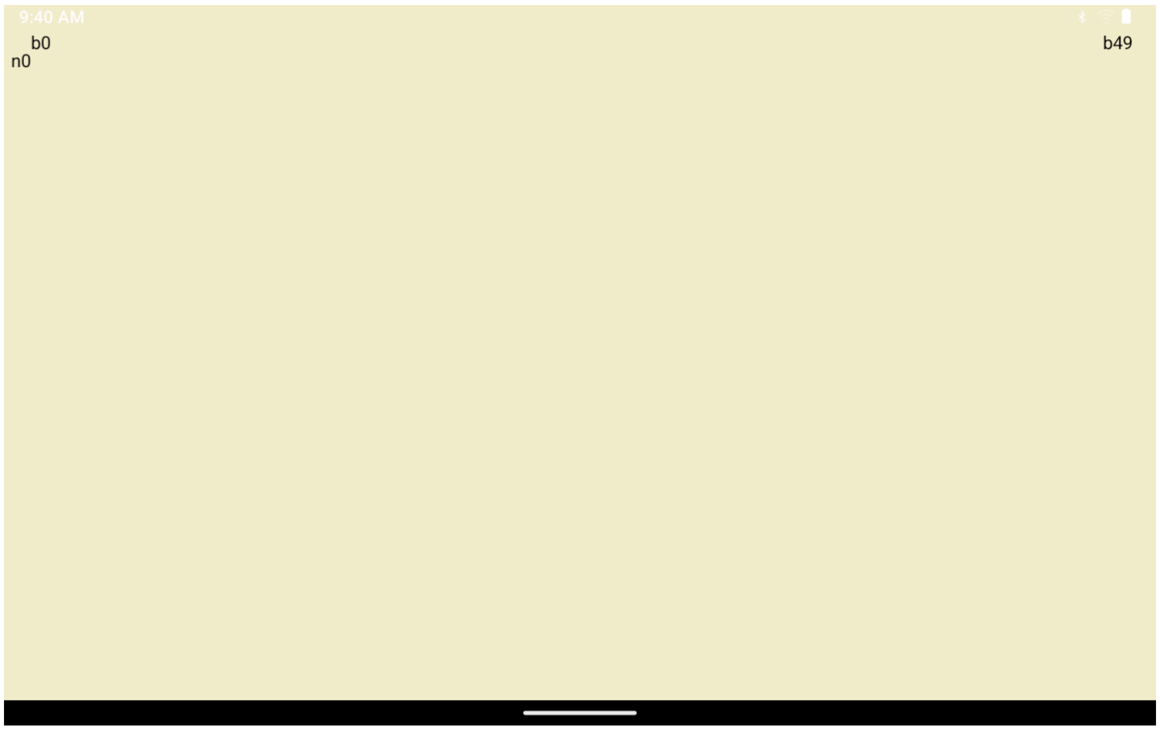


Figure 15: Android Base-2 Write App

The Android base-2 write app is for receiving base-2 data from our machine. Because our reconstruction can only hold two equations at a time, you need to be able to store intermediary computation equations elsewhere. The original used letter-sized paper, and ours uses Android apps. Our machine sends the base-2 data into an ESP32, and the ESP32 transmits that data to the app. Once all of the data is received, the write app sends the page to the read app.

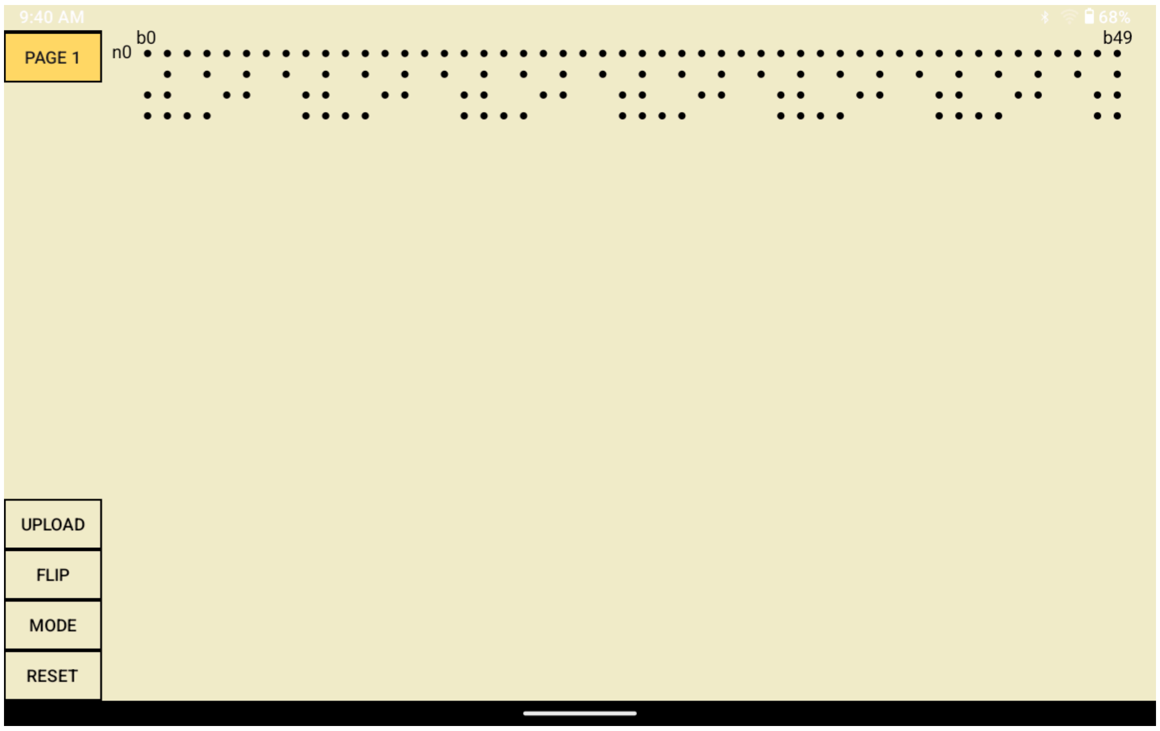


Figure 16: Android Base-2 Read App

The Android base-2 read app is for uploading data back into our machine. The data that is sent from the write app appears as a new page on this app. The data from the read app can be uploaded to an ESP32 that stores the data into the RAM of our machine. The read app also has several quality-of-life options. Users can flip the bits from left to right to make it appear in a readable order. The user can also switch between three modes: dots, 1's and 0's, and base-10. This allows the user to easily read the data as they please. When the pages are no longer needed, the user is able to reset the app back to no pages.

The Android apps, in conjunction with ESP32s, work well and match our final design as planned.

ABC:

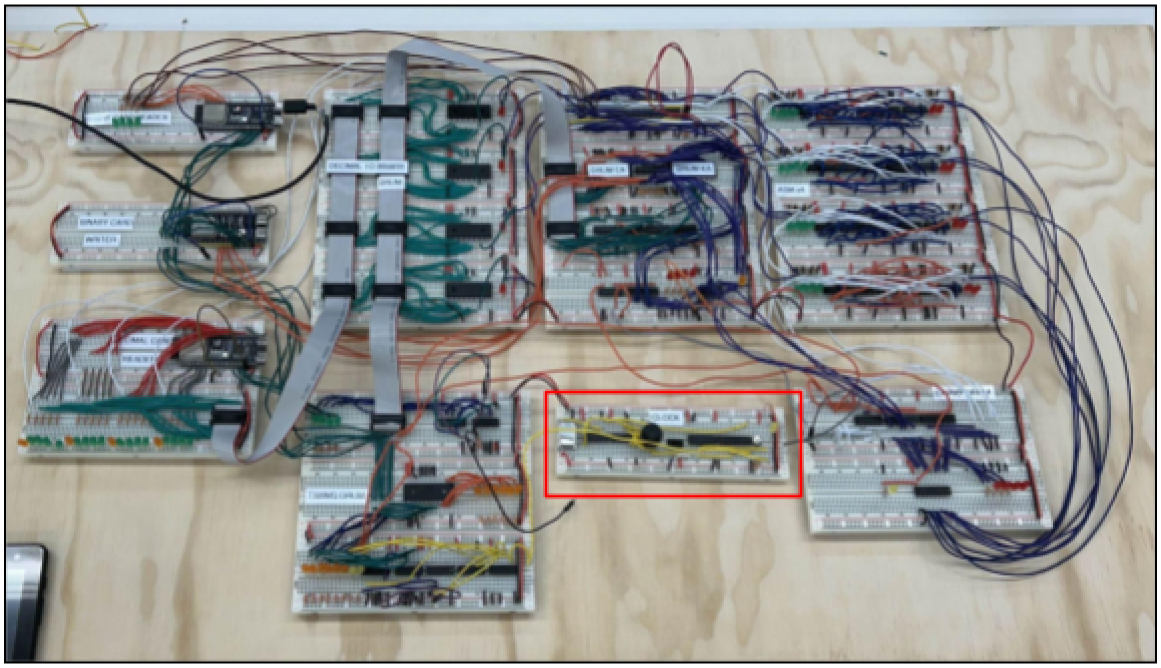


Figure 17: Clock Generator

In moving our design from Quartus simulation to real components on a breadboard, it was necessary for us to first design a clock generator circuit that could provide both an automatic clock with a variable frequency and a manual clock for debugging. Using a 2MHz full can crystal oscillator alongside two clock divider chips that each provide up to 12 outputs of different speeds, a simple adjustable clock circuit was created. Then, by using a button debounced with a register chip, a manual clock was created, and either can be selected via the switch in the middle.

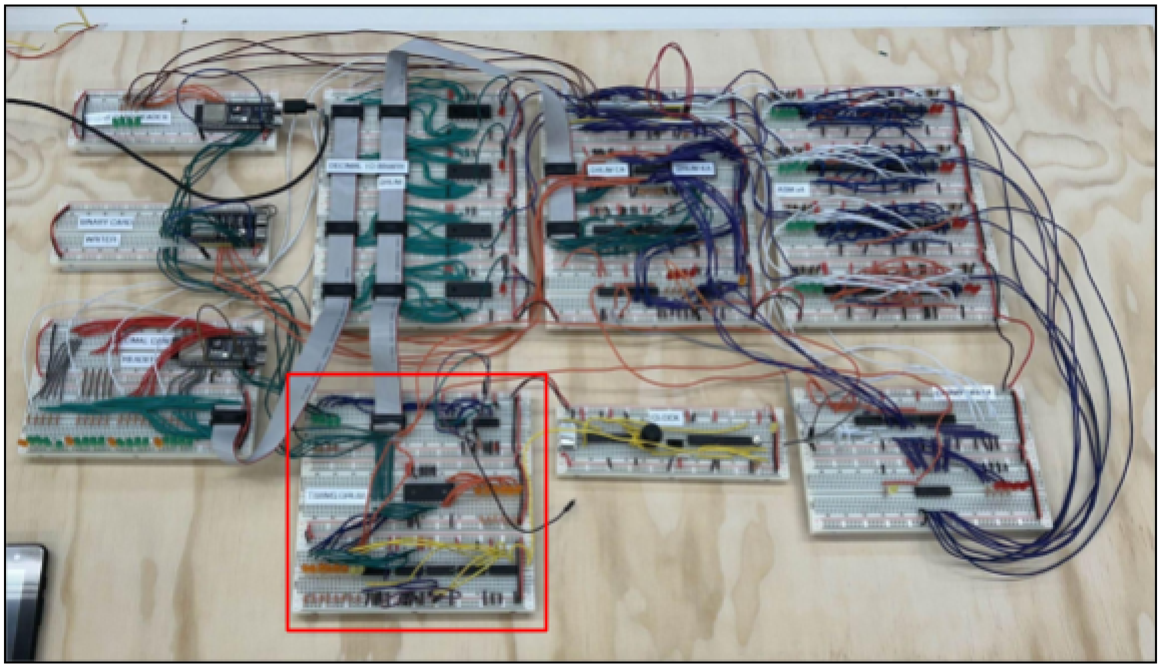


Figure 18: Timing Drum

As above, the timing drum was realized with two, four-bit synchronous reset counters strung together to create a six-bit counter that resets upon reaching the 60th position. This six-bit address is then passed into an EEPROM programmed to output a series of timing signals at each of the addresses to control other parts of the circuit depending on the rotation position. Finally, the board at the top of this module contains a four-bit down counter used when reading in decimal punch cards to count through each of the decades from 14 down to 0.

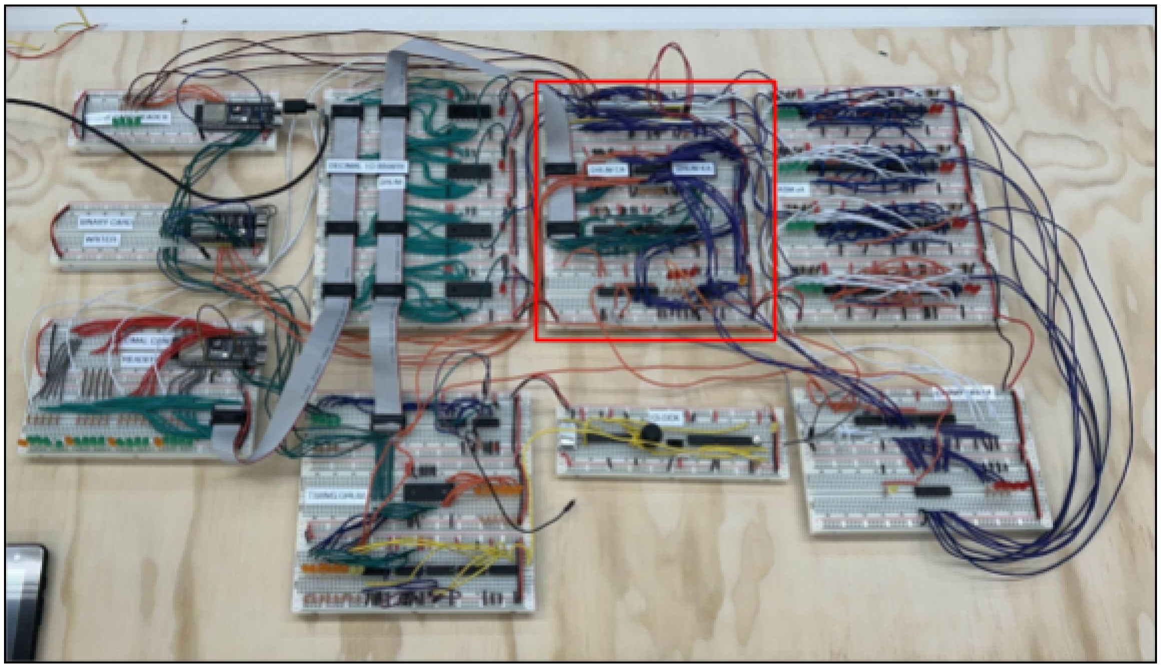


Figure 19: Memory Drums

Here, the two RAM chips for the KA and CA drums are realized with an I/O buffer between them to control the flow of data to and from the RAM chips. The board above the RAM chips contains three multiplexers used to select different data inputs into the ASMs, while the board below calculates a shifted address for the KA drum if a rightward shift is desired.

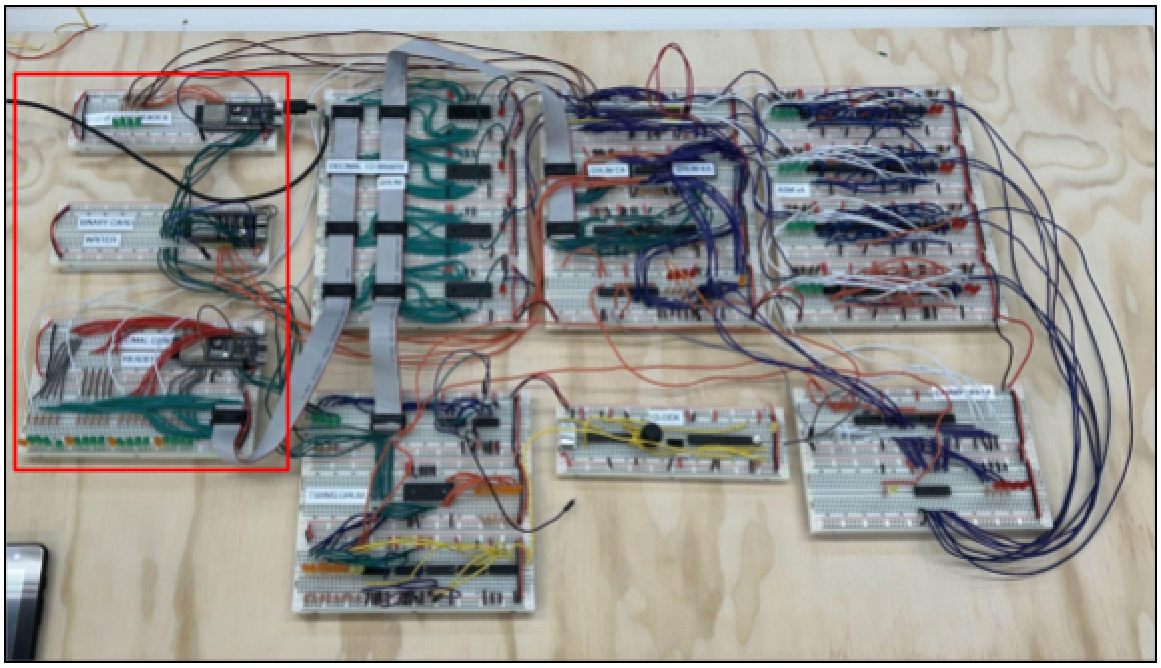


Figure 20: Base-2 Read/Write and Base-10 Write

These boards contain three separate ESP32 microcontroller development boards that control the flow of data into and out of the machine for the punch cards. Since each punch card, both decimal and binary, is being stored on Android tablets, the ESP32 devices provide a method of communication between the tablets and the rest of the machine. The top board is responsible for sending a binary punch card into the machine, while the middle board can read the binary data from the machine to write a binary punch card. Finally, the bottom board is responsible for sending digit data from decimal punch cards into the decimal to binary circuit for entry into the machine as a binary number.

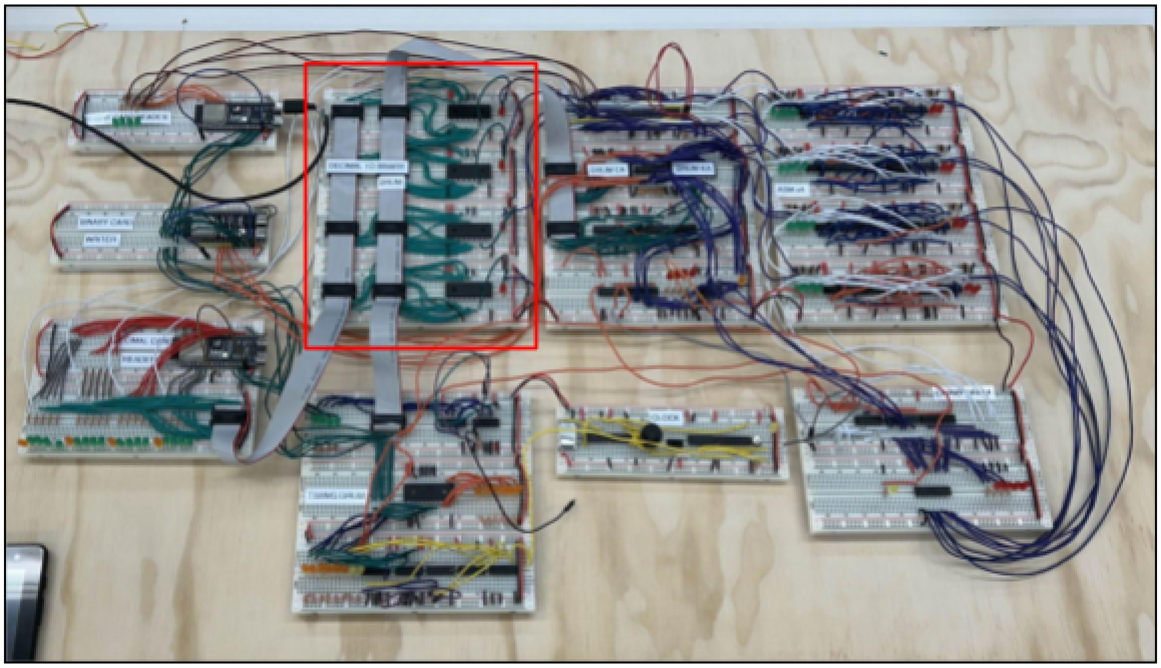


Figure 21: Decimal to Binary Drum

Taking the digit data from the decimal card reader board, the decade number from the four-bit down counter, and finally the six-bit rotational position from the timing drum, the decimal to binary drum computes the proper sequence of bits for converting decimal numbers into binary. This computation is done via four parallel EEPROMS that all receive different digit data from the decimal card reader, and connect into the machine via the ASM multiplexers as described before.

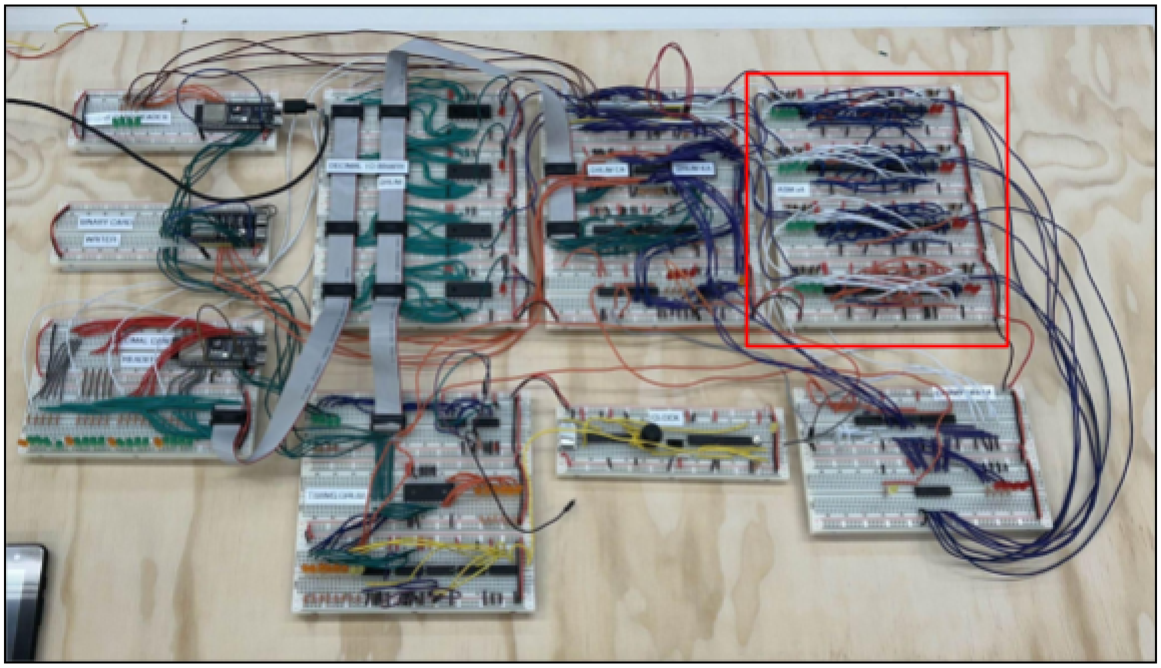


Figure 22: Add/Sub Mechanisms

Each of these boards contains the same digital circuit for performing addition and subtraction operations between two numbers. These circuits are implemented according to the original vacuum tube design converted into modern digital logic gates. While one input of each of these modules always comes from the CA drum, the other can be selected via multiplexers to go between the KA drum, the binary card reader, the decimal to binary drum, and a constant zero. The sum and carry outputs are then stored in the carry drum for use on the next clock cycle.

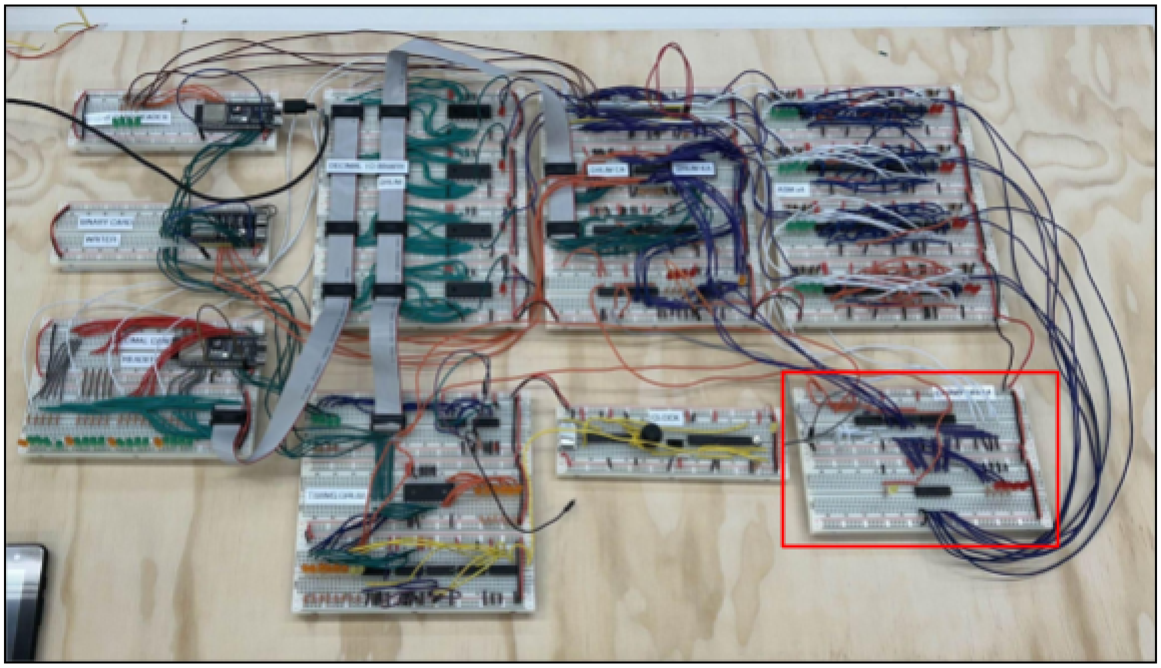


Figure 23: Carry Drum

Finally, the carry drum, which in our implementation stores both the carry and the sum unlike the original, contains a set of registers to hold the sum and carry data from the ASMs for use on the next clock cycle when the sum will be written back to the CA drum while the carry will hold for the next operation.

We are still working to complete the control system and 7-segment display base-10 output. We have been working hard to fully complete our implementation. The reason that these subsystems are still incomplete is the large amount of research we had to complete before being able to design our system, and the amount of time it takes to build the circuits and debug any problems.

abc-emulator:

The abc-emulator is a digital scale model of the Original Atanasoff-Berry Computer written completely in Java. Nothing spins or sparks like the real ABC capacitors and rotating drums. Instead, Java objects hold states, and methods mimic the physical steps.

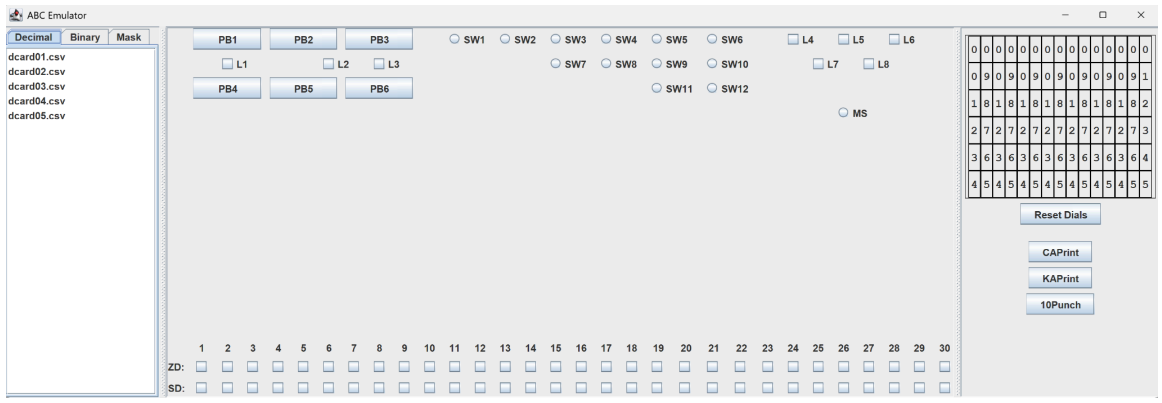
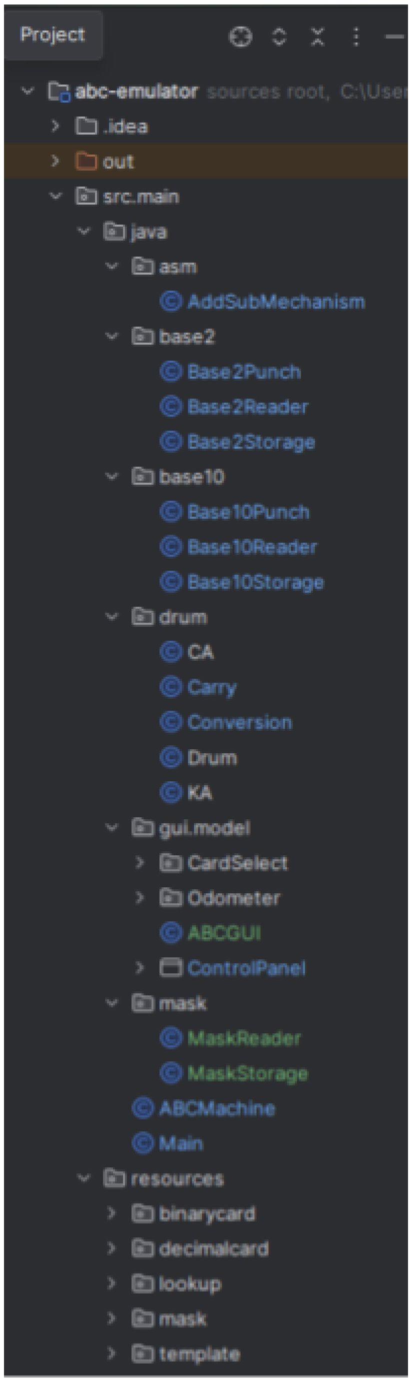


Figure 24: ABCGUI Frame

For the front-end of the simulation, we used Java Swing to recreate the control panel of the original machine. The ABCGUI frame is a split pane layout that holds three panes. CardSelect pane - lists all saved cards, click and push a button to load them. ControlPanel pane - contains the original push buttons, switches, lights, and jumpers, with respective operations, that the original ABC had. The OdometerPanel pane is a six-row by fifteen-column display that displays the value of the band in decimal.



```
Project
abc-emulator sources root, C:\User
  .idea
  out
  src.main
    java
      asm
        AddSubMechanism
      base2
        Base2Punch
        Base2Reader
        Base2Storage
      base10
        Base10Punch
        Base10Reader
        Base10Storage
      drum
        CA
        Carry
        Conversion
        Drum
        KA
      gui.model
        CardSelect
        Odometer
        ABCGUI
        ControlPanel
      mask
        MaskReader
        MaskStorage
      ABCMachine
      Main
    resources
      binarycard
      decimalcard
      lookup
      mask
      template
```

Figure 25: abc-emulator Project Structure

Core emulator back-end:

- Drum data
  - Has subclasses CA, KA, and Carry.
  - CA and KA have 30 bands by 50 bit boolean matrix, with bitwise getter, setters, clears, and shift operations.
- Add/Sub Mechanism (ASM)
  - Contains boolean full-adder logic (addSubBit), with band processing and equations loops matching Stoytchev's "Key Steps"
- Card I/O
  - Decimal, Binary, and Mask cards are all saved as csv files. With Reader, Storage, and Punch classes for each. Files are stored in auto-incrementing filenames (dcardXX.csv, bcardXX.csv) and on-disk caching under resources/... .
- Conversion Drum
  - Is a lookup table for Base10 Reading. Loads lookup/convdrum.csv into a memory boolean and performs the brush-offset lookup that translates decimal digits into 50 bit binary boolean arrays.

How it matches the final design:

The objectives that were met in the abc-emulator implementation include. Emulating CA, KA, Carry, and Conversion drums at the bit level. The Drums mirror this with 30 by 50 MSB-LSB boolean arrays. Implement Berry's seven-step elimination algorithm. This is fully coded in AddSubmechanism.operation. Punched cards and allow re-loading. This is met with Decimal, Binary, and Mask storages, reading and punching. Functional control panel is met with the three UI panes. A part that wasn't me was simulation of drum rotation and timing. Was not implemented.

Features, functions, sub-systems that are implemented:

Bit accurate drums (CA, KA, Carry) have respective operations that as a team we researched. The csv lookup to convert decimal to binary. Full Add/Sub Mechanism that matches the original logic Atanasoff used. Decimal, Binary, Mask cards. Three-pane Swing GUI with live states. Enums and toggle logic for Switches and Pushbuttons relating to the original machine.

Did not finish and why:

Accurate drum rotation was not met because it adds a large amount of threading complexity. Also allowed us to focus on low-level implementation.

6.1 Design Analysis

As stated before, the primary goal of our design was to provide an adequate balance between staying authentic to the original design of the ABC and deviating from that in favor of modernity and understandability. Specifically, we made three critical design decisions in favor of sticking with the original design to preserve our connection to its legacy, and we believe that within the context of those decisions, we have found a sufficient design and implementation.

We understand that parts of our design may seem inefficient or suboptimal when compared to modern computing standards, however, we firmly believe that any more diversions away from Dr. Atanasoff and Berry's original design and in favor of efficiency or modern technology would ultimately sacrifice too much of the historical and spiritual connection that our design is supposed to have with its predecessor. To this aim, we believe that our design is rather successful.

7. Ethics and Professional Responsibility

7.1 Areas of Professional Responsibility/Codes of Ethics

Area	Definition	Relevance to Project
Work Competence	While working on the project we are outputting high quality, integrity, timeliness, and professional competence.	Our skill sets covered the needs of our project. With Electrical, Computer, and Software Undergraduate Engineers populating our team, we were able to work in areas of our competence.
Financial Responsibility	Order parts that are needed to complete our project.	Our implementation was mainly breadboarded and software based. We ordered chips through ETG and put our Amazon order through them. Our team would go through a parts order after Friday meetings to make sure we are getting what we need to build our project. We only had one PCB order from JLCPCB to create our Add-Sub Module. By sitting down with each other and going through together we spent money responsibly.
Communication Honesty	Attend weekly meetings, communicate through discord, and be honest through work.	We met twice a week where we updated our weekly report, filled out design documents, and took meeting notes. All of our updates are truthful in what we have done each week and the progress we've made. We communicate fully with our advisor and constantly update each other.
Health, Safety, and Well-being	Reduce risks of safety, and health.	If a member was sick we recommended staying home and not forcing attendance in a meeting. We also used best practices when breadboarding and implementing other parts. By adhering to this area, we built confidence in our project and reduced sick time.
Property Ownership	Respect each other's ideas, and release our project to the public.	Our final product will be released under a permissive open-source license. We acknowledge our own team's contributions by printing names of the student(s) on each PCB that they worked on. We also have a team logo that will be printed on each PCB.
Sustainability	Use our school's resources.	Our team ordered parts from China, through ETG. That way, we stay under budget and get fast, efficient PCBs and chips. We also placed Amazon orders through the school to get parts for our final design.
Social Responsibility	Design an educational product that provides learning.	To us, this means serving a societal purpose that can inspire creativity in people interested in computers.

Figure 26: Areas of Professional Responsibility and Their Relevance to Our Project

The area where our team performed well is Social Responsibility. One of our team's virtues is staying faithful to the ABC's original logic, just interpreted in a modern form. By staying true to this, we created a meaningful product. We want people interested in computers and students in computer-related degree programs to be able to easily understand how the first electronic computer operated.

One area our team needs to improve on is Sustainability. We didn't think heavily on optimization of an algorithm because we're using an already existing one. Also, we ordered parts from China because it's fast and cheap. This is done because we're under a time constraint and do not have the luxury of spending a large amount of money.

7.2 Four Principles

Area	Description
Beneficence	Our project respects the values of people in computer related fields and people interested in computers. We reflect their values by attempting to stay as faithful as we can to the original design.
Nonmaleficence	Relating this area to the environment, we find some problems. We order parts from China because they are cheap, but we're not sure how friendly their practices are to the environment. We don't think twice about being supplied PCBs from China.
Respect for Autonomy	We are releasing our project under a permissive open-source license. This allows people invested in the project to modify our project themselves on their own account. This promotes creativity and allows for future teams to learn and base on our project.
Justice	With an Economic relation, we promote fairness by standardizing the parts we order. We conversed with the other team that Stoytchev, our advisor/client, is also supervising, and we standardized LED's and other wiring's for our prototypes.

Figure 27: Broader Context Principle Pairs

An important pair is Beneficence-Global, Cultural, and Social. This is important because our team wants to deliver a product that can be benefitted by students and anyone interested in computers. We will ensure it by considering multiple perspectives by breaking down the basic principles of the original ABC. A pair that is lacking is Nonmaleficence-Environmental. The main negative of this pair is that we have little to zero control over this. We work under Iowa State and use the basics. This is overcome in other areas because it's so insignificant to our entire project.

7.3 Virtues

Team Virtues

- Collaboration
    - Ensures that team members work together, this helps each other's strengths to achieve a common goal. This promotes creativity by combining different perspectives.
    - As a team we met twice a week and communicated fully through Discord. We directed each other's roles to each of our strengths in completing our project. By doing so we were able to positively collaborate.
- Integrity
    - Integrity builds trust in our team and with our client. It ensures that decisions are made ethically and that work is performed honestly and transparently.
- Adaptability
    - Adaptability allows our team to respond effectively to unexpected challenges, such as changing project requirements, technical issues, or new insights gained during prototyping.

8. Conclusions

8.1 Summary of Progress

The team has developed a full design of the ABC re-construction and we have nearly implemented the entire design. The Android apps used for input/output are finished and working. Most of the machine's parts, except for the control unit and 7-segment display output, are finished. We are still working hard to fully complete our implementation before our presentation.

8.2 Value Provided

Our design address's our user needs and it fulfills all requirements that we set out to fulfill. In the grand scheme of things, our design could be helpful for future students to understand how computational logic works. It also provides an important historical understanding of how computers became what they are today. The ABC being a simple computer makes it possible for students to understand what it is doing. The LEDs visualizing each step are also a huge help to understanding the inner-working of the computer.

8.3 Next Steps

Our project could go in a lot of different ways from here. We would like to see another team take our design and fully implement it on PCBs. This would greatly decrease the size of the implementation and could even allow the team to develop a full-fledged 30 coefficient version of the ABC. Another idea for future projects is to start developing the ABC 100% faithfully. This would take multiple teams. For example, one team could develop the base-2 puncher and reader circuits while another team develops the add-subtract mechanisms. We are proud to be the beginning of what could be some great projects.

9. References

[1] A. Burks and A. W. Burks, "The First Electronic Computer: The Atanasoff Story." Ann Arbor: University of Michigan Press, 1988.

[2] IEEE Standard Test Access Port and Boundary-Scan Architecture, IEEE Std 1149.1-2001.

[3] IEEE Standard for Integrated Circuit (IC) Designer Metrics, IEEE Std 1481-2009.

[4] IEEE Standard for User Interface Elements in Power Control of Electronic Devices, IEEE Std 1621-2004.

[5] IEEE Standard for Environmental Assessment of Personal Computer Products, Including Laptop Personal Computers, Desktop Personal Computers, and Monitors, IEEE Std 1680-2006.

[6] IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2019.

[7] J. Gustafson and C. Shorb, "The Atanasoff-Berry Computer In Operation," YouTube, https://www.youtube.com/watch?v=YyxGIbtMS9E&ab_channel=ComputerHistoryMuseum (accessed 2024).

[8] J. Gustafson, Reconstruction of the Atanasoff-Berry Computer, http://www.johngustafson.net/pubs/pub57/ABCPaper.htm (accessed 2024).

10. Appendices

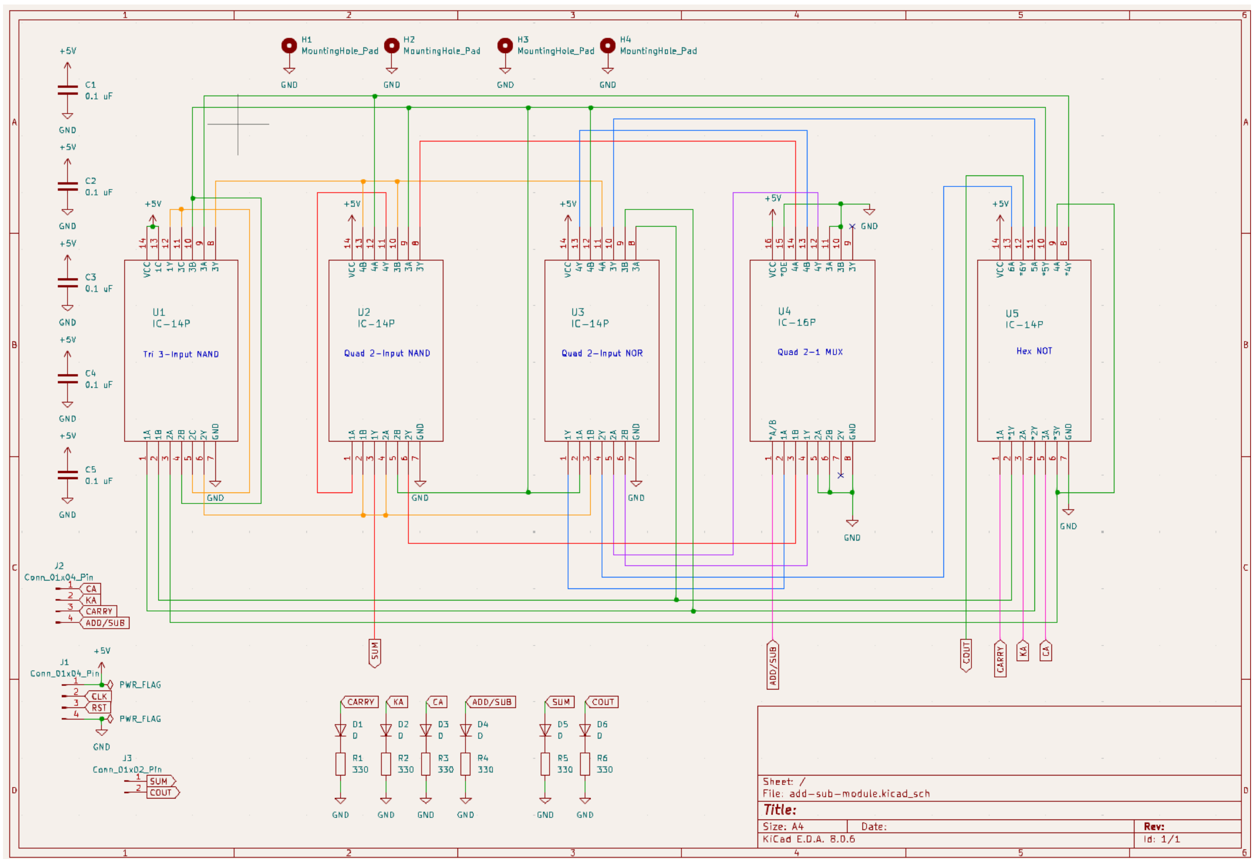


Figure 28: Adder-Subtractor Module KiCad Schematic

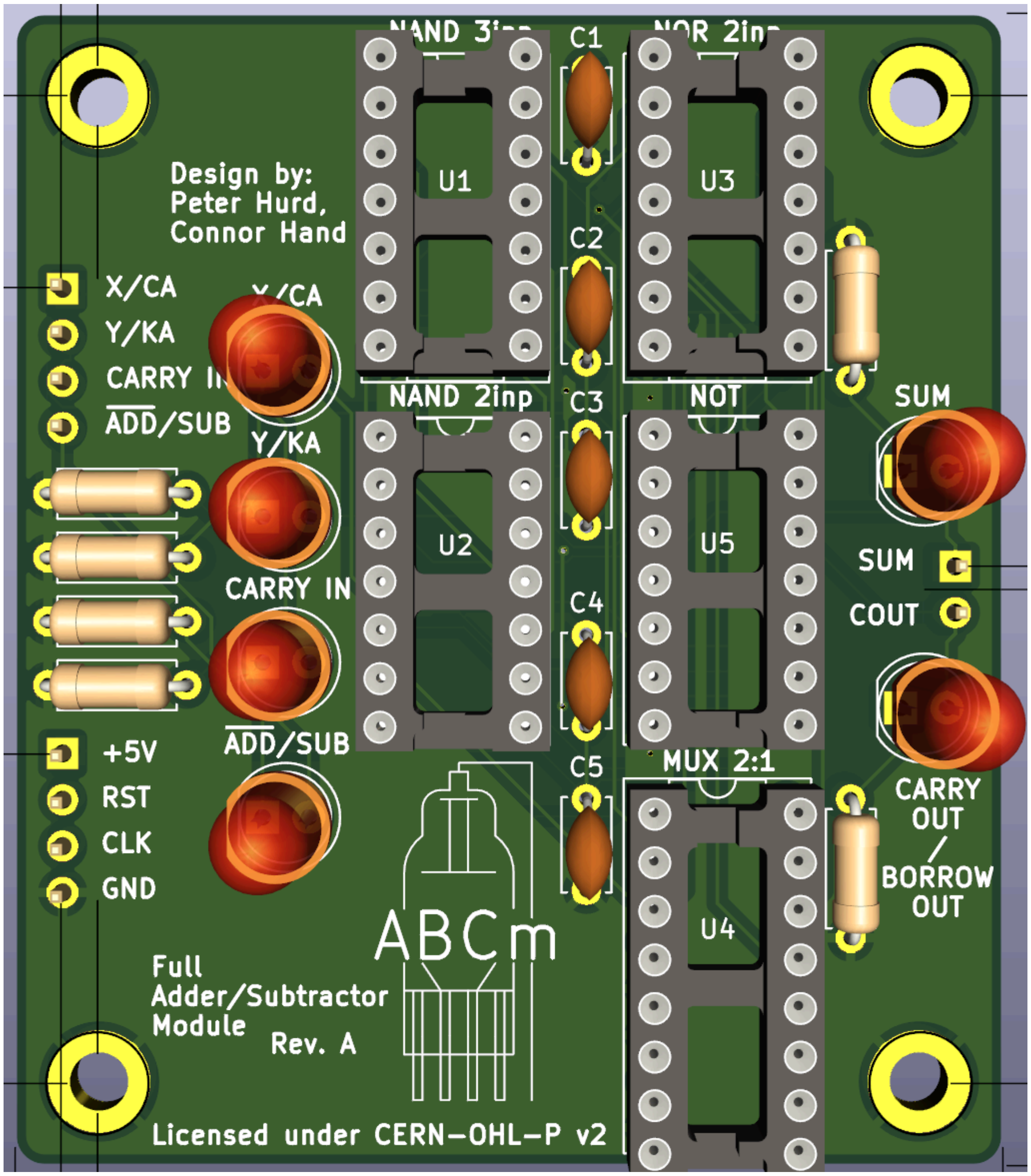


Figure 29: Adder-Subtractor Module PCB Implementation Render

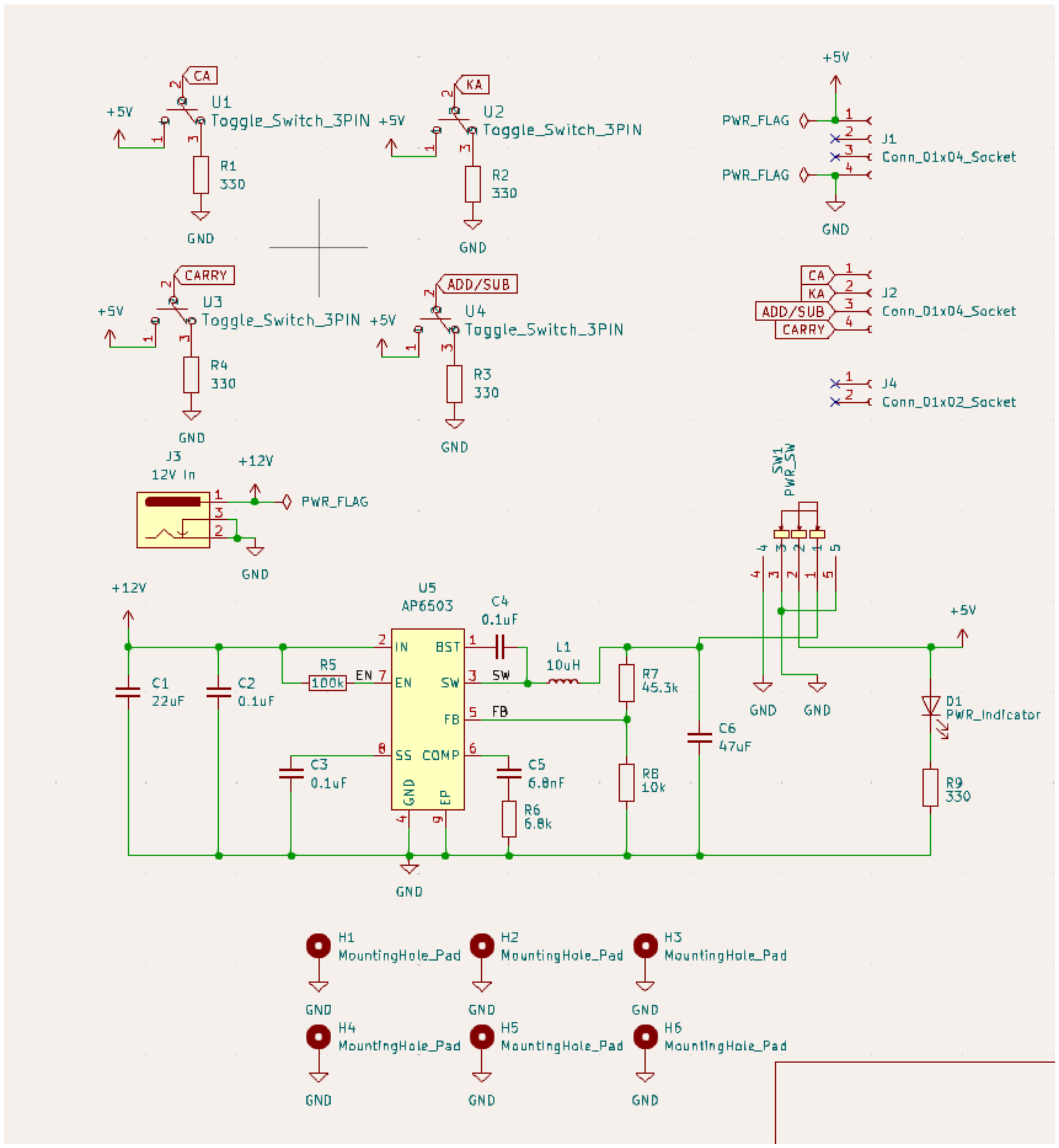


Figure 30: KiCad Schematic for the Adder-Subtractor Tester PCB

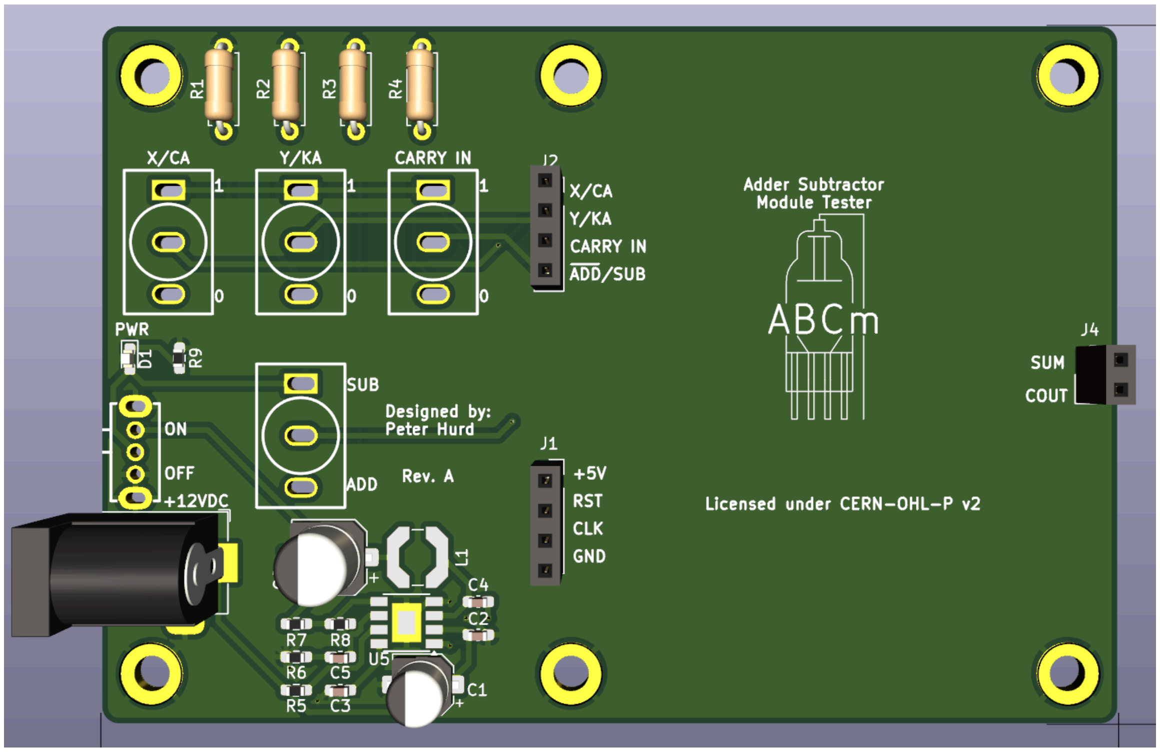


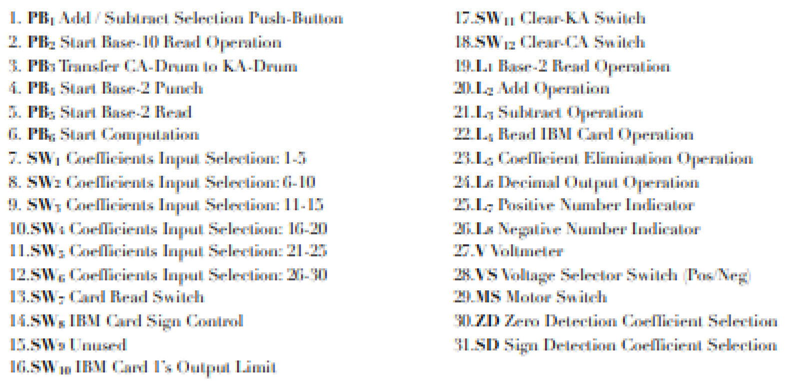
Figure 31: Adder-Subtractor Tester PCB Implementation Render

Figures 13-16 show the PCB designs that we created when we thought we were going to do a full PCB implementation. The adder subtractor PCB was for adding and subtracting binary numbers. The adder subtractor tester was specifically designed to test the adder subtractor PCBs using switches and LEDs.

Appendix 1 - Operation Manual

Note: Our operation manual goes over the use of the abc-emulator Java simulation as our physical implementation is not complete enough to fully use yet.

Legend:



1. **PB$_1$** Add / Subtract Selection Push-Button
2. **PB$_2$** Start Base-10 Read Operation
3. **PB$_3$** Transfer CA-Drum to KA-Drum
4. **PB$_4$** Start Base-2 Punch
5. **PB$_5$** Start Base-2 Read
6. **PB$_6$** Start Computation
7. **SW$_1$** Coefficients Input Selection: 1-5
8. **SW$_2$** Coefficients Input Selection: 6-10
9. **SW$_3$** Coefficients Input Selection: 11-15
10. **SW$_4$** Coefficients Input Selection: 16-20
11. **SW$_5$** Coefficients Input Selection: 21-25
12. **SW$_6$** Coefficients Input Selection: 26-30
13. **SW$_7$** Card Read Switch
14. **SW$_8$** IBM Card Sign Control
15. **SW$_9$** Unused
16. **SW$_{10}$** IBM Card 1's Output Limit
17. **SW$_{11}$** Clear-KA Switch
18. **SW$_{12}$** Clear-CA Switch
19. **L$_1$** Base-2 Read Operation
20. **L$_2$** Add Operation
21. **L$_3$** Subtract Operation
22. **L$_4$** Read IBM Card Operation
23. **L$_5$** Coefficient Elimination Operation
24. **L$_6$** Decimal Output Operation
25. **L$_7$** Positive Number Indicator
26. **L$_8$** Negative Number Indicator
27. **V** Voltmeter
28. **VS** Voltage Selector Switch (Pos/Neg)
29. **MS** Motor Switch
30. **ZD** Zero Detection Coefficient Selection
31. **SD** Sign Detection Coefficient Selection

Figure 32: Legend for abc-emulator Buttons and Switches

Control Panel UI:

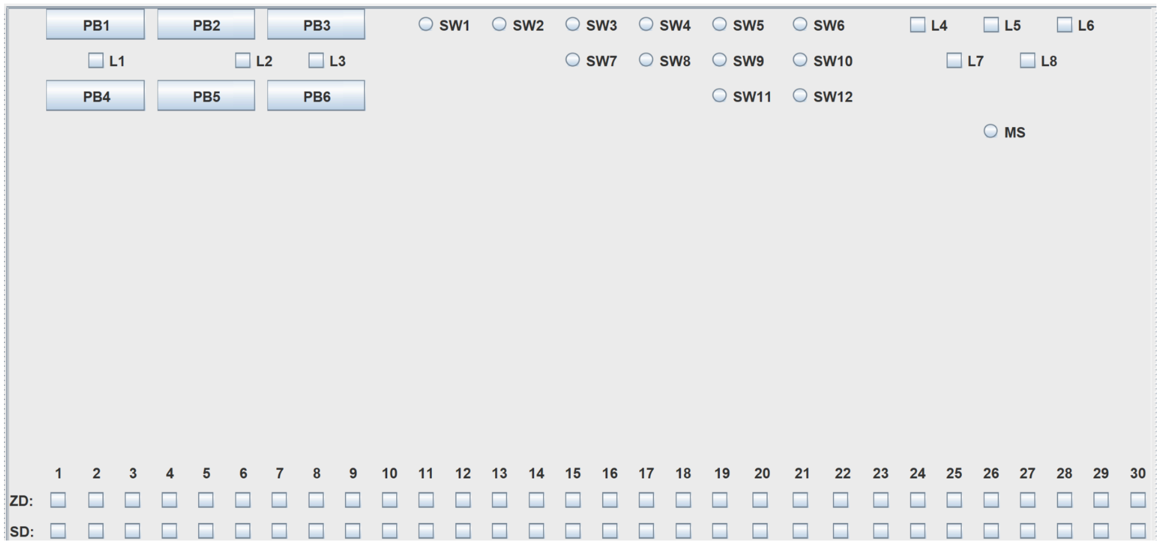


Figure 33: abc-emulator Control Panel

Reading Base-10 Cards:

1. Turn power switch MS on.
2. Set switch SW7 so that inputs to ASM's are connected to card reader.
3. Set switch SW8 so that add-subtract relay is controlled by card (rather than by sensing overdrafts).
4. Select ADD with pushbutton PB1, by monitoring the lights L2 and L3.

5. Select field on CA into which card is read by closing one of the six switches SW1-SW6.
6. Clear CA and KA with switches SW11 and SW12 respectively.
7. Plase first Base10 card in reader (Equation to be eliminated goes in first).
8. Press button PB2 to cause base-10 card reader to GO.
9. For more than five coefficients, select new field by closing the next switch SW1-SW6, add a new card, and repeat until all coefficients of one equation are in CA.
10. Transfer contents of CA to KA by pressing GO button PB3.
11. Repeat above to place all coefficients of second equation on CA.

Computing:

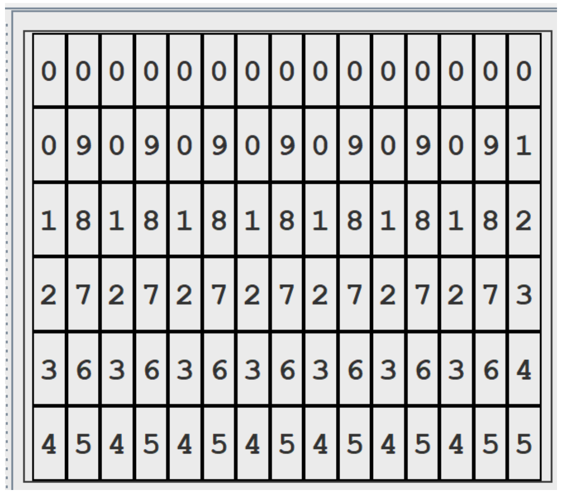
12. Set switch SW7 so that inputs to ASM's are connected to KA.
13. Select coefficient to be eliminated by selecting checkbox ZD .
14. Set switch SW8 so that add-subtract relay will be switched by overdrafts.
15. With pushbutton PB1, set add-subtract control to ADD if signs of coefficients to be eliminated were alike - set to subtract if they are the same.
16. Press Go Compute button PB6 (computation begins on next cycle and is auto-matically terminated when equation is reduced to zero.)
17. Press pushbutton PB4 to punch contents of CA (the answer) onto a base-2 card.

Base-2 Reading:

18. Return SW7 to "card read" position.
19. Place machine in ADD by pressing PB1 if necessary.
20. Place first card in Base-2 reading rack.
21. Clear CA and KA by pressing SW11 and SW12 respectively (optional step).
22. Read first card into CA by pressing button PB5.
23. Transfer contents of CA to KA by pressing button PB3.
24. Clear CA by pressing SW11.
25. Place second card in Base-2 reading rack.
26. Read second card into CA by pressing button PB5.

Base-10 Reading:

27. Select checkbox SD at coefficient location to be read.
28. Check sign of number using L7 and L8 for positive and negative number indicator (respectively).
29. Place machine in SUBTRACT if number to be read is positive (in ADD if negative) by means of button PB1.
30. Open switch SW10 to connect only powers of 10 brush in base-10 card reader to ASM, making sure no IBM card has been left in base-10 card reader.
31. Select mask card for coefficient to be read.
32. Set switch SW7 to card read position.
33. Set switch SW8 to sense overdrafts.
34. Press the Compute "Go" button PB6.
35. Read base-10 number on odometer panel after computation is complete.



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	9	0	9	0	9	0	9	0	9	0	9	0	9	1
1	8	1	8	1	8	1	8	1	8	1	8	1	8	2
2	7	2	7	2	7	2	7	2	7	2	7	2	7	3
3	6	3	6	3	6	3	6	3	6	3	6	3	6	4
4	5	4	5	4	5	4	5	4	5	4	5	4	5	5

Figure 34: Odometer display for reading Base-10

Appendix 2 - Initial Version of Design

Initially, we planned on using printed circuit boards for our final implementation. This was scrapped due to there not being enough time for us to achieve this goal. This was partly due to the fact that researching and creating a design took significantly longer than expected.

Appendix 3 - Other Considerations

- The reconstruction team from the 90s did not leave much of anything, except for a small demo video, for us to use in terms of resources. We aren't even sure if it worked exactly like the original.
- Our best sources were legal documents from the original.

Appendix 4 - Code

GitHub Repo: https://github.com/phurd22/sdmay25-29

Appendix 5 - Team Contract

Team Members

- Peter Hurd
- Connor Hand
- Zach Scurlock
- Noah Butler
- William Mayer

Required Skill Sets for Your Project

The required engineering students for the implementation of ABC with modern technology are: Electrical, Computer, and Software. Some other special skills needed include experience with breadboards, familiarity with computer design principles, and programming skills.

Skill Sets Covered by the Team

- Electrical Engineering
  - Peter Hurd
- Computer Engineering
  - Connor Hand
  - Zach Scurlock
  - Noah Butler
- Software Engineering
  - William Mayer

Peter Hurd, Connor Hand, Zach Scurlock, and Noah Butler all have prior experience with breadboards. William Mayer has programming skills. All team members are familiar with computer design principles.

Project Management Style Adopted by the Team

Our team uses a mix of both Waterfall and Agile styles for our project management. As a team, we defined our project's scope, objectives, and requirements. We set milestones with a project timeline. From there, we broke down our work into short sprints, focusing on specific tasks. We often met and gathered feedback from our advisor to make flexible adjustments.

Individual Project Management Roles

1. Project Sponsor
   a. Iowa State University
2. Project Manager
   a. Alexander Stoytchev
3. Project Team Members
   a. Connor Hand
      i. Client Interaction and Team Organization
   b. Zach Scurlock

        i. Testing and Individual Component Design

   c. Peter Hurd
        i. Testing and Individual Component Design / Budget Handling
   d. William Mayer
        i. Meeting Tracking and Note-Taking
   e. Noah Butler
        i. Testing and Individual Component Design

Team Contract

Team Members:

1) Connor Hand 2) Zach Scurlock

3) Peter Hurd 4) Noah Butler

5) William Mayer

Team Procedures

1. Day, time, and location for regular team meetings:

With advisor - Thursday at 1:30 PM Senior Design Rm in Coover

Without advisor - Wednesday at 3:30 PM Senior Design Rm in Coover

2. Preferred method of communication updates, reminders, issues, and scheduling:

Discord - Outlook

3. Decision-making policy:

Majority Vote

4. Procedures for record keeping:

William records notes - Shared in Google Drive and Notes channel

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

All individuals should be attending at least 90% of meetings. We understand some situations are unavoidable. Team members should participate in meetings by sharing what they have done since the last meeting and by helping with the project.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

Assignments: Get done before the due date

Timelines: Try your best for timelines

Deadlines: Finish up before deadlines

3. Expected level of communication with other team members:

Team members should respond to one another within the same day. Team members should be sharing important information with each other or asking each other questions over Discord. Expect responses at a reasonable time of day.

4. Expected level of commitment to team decisions and tasks:

Complete other school work but give full commitment when possible.

Leadership

1. Leadership roles for each team member:

Client Interaction / Team Organization: Connor Hand

Meeting Time Tracking / Note Taking: William Mayer

Testing / Individual Component Design: Noah Butler, Zachary Scurlock, Peter Hurd

Budget Handling: Peter Hurd

2. Strategies for supporting and guiding the work of all team members:

Constant communication

Ask for help whenever

Rubber Ducky theory

3. Strategies for recognizing the contributions of all team members:

Time management

Have responsibility for team actions

Talk about what we did individually at each meeting

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

Connor Hand: Hardware design, some electronic circuits, computer theory

Zachary Scurlock: Hardware design, computer theory, software design

Peter Hurd: Electronic circuits, computer theory

William Mayer: Software concepts, multiple coding languages

Noah Butler: Hardware design, software design

2. Strategies for encouraging and supporting contributions and ideas from all team members:

We are using Discord for our communication so we can put our ideas in our Discord server, where they will be forever and everyone can see. We can also talk about our contributions and ideas during our meetings.

3. Procedures for identifying and resolving collaboration or inclusion issues:

By having an open-minded conversation. We all agree that our meetings are a safe space to talk about any problems we have within the team.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

Develop a functional, final implementation of our design.

2. Strategies for planning and assigning individual and teamwork:

Determine work to be done at meetings and create a list of items that need to be completed. We can take the items that we want to do or the team has agreed on and assign our names to them.

3. Strategies for keeping on task:

Meet consistently and communicate constantly. Create personal goals to spend a certain amount of time on our project individually.

Consequences for Not Adhering to Team Contract

Open-minded conversations and heavy communication if infractions repeat. Communication with Stoytchev on specifics of infractions. If infractions continue after talking about it with the team, the rest of the team and Stoytchev will determine the next steps for the person committing the infractions.

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the

consequences as stated in this contract.

1) Connor Hand	DATE 5/4/2025
2) Zachary Scurlock	DATE 5/4/2025
3) William Mayer	DATE 5/4/2025

4) Noah Butler	DATE 5/4/2025
5) Peter Hurd	DATE 5/4/2025