

# AUTOMATED IC CHIP TESTER

By

Michael Ruscito

Alison Shikada

Ryan Yoseph

Final Report for ECE 445, Senior Design, Spring 2021

TA: Anand Sunderrajan

5 May 2021

Project No. 47

## Abstract

We designed and built a device for students in ECE 385 to automate the process of unit testing IC chips. Students place their IC chip in the socket on our printed circuit board and then select the chip they want to test on the website that is hosted by our project. The web application then communicates the user's selection with our hardware which runs a series of test vectors comparing the measured outputs from the user's IC chip to the expected outputs and determines whether the chip is working properly or not. The device then relays this result to the user in under two milliseconds. The device is designed to provide the user with a simple user interface and an easy solution for IC chip testing in the ECE 385 lab.

## Contents

1. Introduction .....	1
1.1 Purpose .....	1
1.2 High-Level Requirements .....	2
1.3 Subsystem Overview .....	2
2 Design .....	4
2.1 Power Supply .....	4
2.1.1 Lithium Polymer Battery .....	4
2.1.2 USB Port .....	4
2.1.3 Power Switch .....	4
2.1.4 Charging Switch .....	5
2.1.5 Power LED .....	5
2.1.6 Charging LED .....	5
2.1.7 3.3 V Low-Dropout (LDO) Regulator .....	5
2.1.8 3.3 V/5 V Synchronous Boost Converter .....	5
2.1.9 Charging Subsystem .....	6
2.2 Control Unit .....	6
2.2.1 ESP32 Microcontroller .....	7
2.2.2 Port Expander .....	7
2.2.3 Tristate Buffers .....	8
2.2.4 2-to-4 Decoder .....	8
2.2.5 LEDs .....	8
2.2.6 USB-to-UART Bridge .....	8
2.3 Wi-Fi Application .....	9
2.3.1 Flowchart .....	10
3. Design Verification .....	11
3.1 Power Supply .....	11
3.2 Control Unit .....	11
3.2.1 Peripheral Hardware .....	11
3.2.2 ESP32 Microcontroller .....	12
3.3 Wi-Fi Application .....	14
4. Costs and Schedule .....	15

4.1 Parts .....	15
4.2 Labor .....	16
4.3 Total Costs.....	16
4.4 Schedule.....	16
5. Conclusion.....	18
5.1 Accomplishments.....	18
5.2 Uncertainties.....	18
5.3 Ethical considerations .....	18
5.4 Future work.....	19
References .....	20
Appendix A Requirement and Verification Table.....	21
Appendix B PCB Layout .....	23
Appendix C Control Unit Eagle Schematic.....	24

# 1. Introduction

## 1.1 Purpose

A common frustration in ECE 385 is when students find a given IC chip does not work as expected. Testing each chip manually is not only tedious, but time consuming. Students are often encouraged to test individual chips before using them, but this advice is rarely followed. By adding the time constraint of assignments, complexity of 385 labs, and the potential to ruin chips, neglecting to unit test the IC chips incorrectly influences students to think that their IC chips are in mint condition.

Our project is targeted toward students by providing a small, portable solution to unit test IC chips quickly and easily. Our goal is to automate the process of chip testing by using a database of TI datasheets and a streamlined User Interface (UI) for easy testing. The user would only need to select the chip model number and press “submit”. Internally, our PCB uses the chip number to identify the appropriate signals and sends them to the IC chip for testing. Our firmware returns a true or false value and lights up the corresponding blue or green LED to indicate the chip’s working condition. Our project consists of a physical device and web application. Figure 1 depicts the physical testing device and battery.

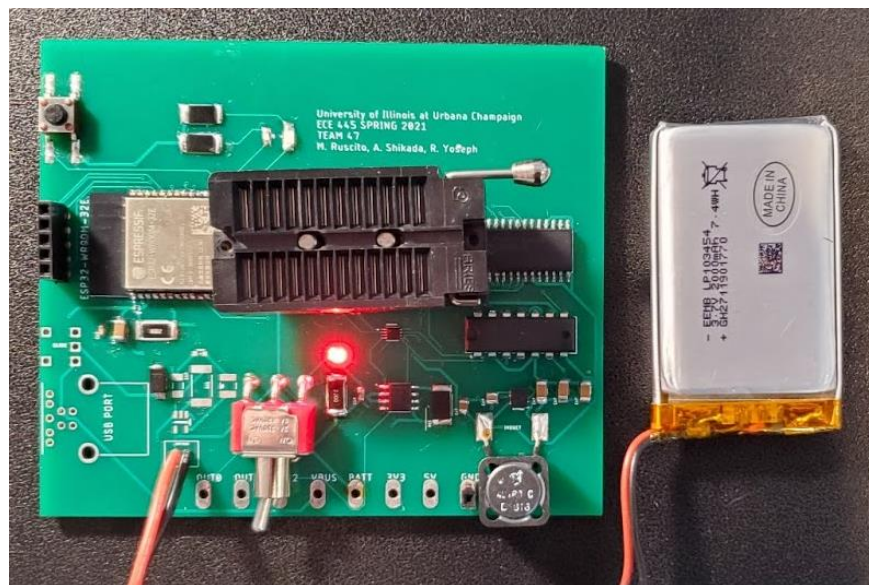


Figure 1: Automated IC Chip Testing Device and Battery

Manufacturers themselves are not necessarily to blame for faulty IC chips. There are several tests run during the IC chip fabrication process: such as Pre-Burn-in, Burn-in and Final Test [1]. Burn-in testing occurs when a device is put under elevated temperatures to “ensure required high quality and reliability of the produced semiconductors before shipping them to final users.”

In his research on “Reliability Challenges in 3D IC Packaging Technology”, Tu [2] concludes that the most serious reliability concern is joule heating, or the passage of electric current through a conductor produces heat. In a class setting, this is especially prevalent. According to Ronen and Eliahu, most of the

common difficulties students have in the study of simple electric circuits are due to an incomplete understanding of the concepts by which idealized models predict the behavior of a system [3]. By offering a portable IC tester, we check the chip using the practice of “idealizing” the circuit model.

## 1.2 High-Level Requirements

Any substantial project needs to be grounded in a set of tangible goals. For this project, our team produced the following high-level requirements that measure performance, breadth, and portability so that we could gauge our success based upon these specific outcomes:

1. The device’s internal logic must be able to determine if a chip is working properly in under 2 milliseconds to provide the user with fast results.
2. The interface between the ESP32 and Wi-Fi device can select from the 18 types of IC chips provided in the ECE 385 standard lab kit and output the correct testing conditions for the chip.
3. The power supply must contain a rechargeable Lithium Polymer (LiPo) battery complete with an integrated battery charging system that grants the user 3 hours of use time.

## 1.3 Subsystem Overview

The device requires three main subsystems to meet the high-level requirements: a regulated power supply, a dedicated control unit, and a smartphone or computer that can host the web application.

The input to the power supply is a rechargeable Lithium Polymer (LiPo) battery. Two different voltage regulators, a Low Dropout (LDO) regulator and DC-DC boost converter, provide Constant Voltage/Constant Current (CV/CC) to the device and test chip. Additionally, a USB cable, combined with a battery management controller, facilitates fast and safe charging of the LiPo battery.

The control unit manages the execution of the test suite for the IC chip under test and does so by providing signals to the ZIF socket and communicating with the smartphone via Wi-Fi. The ZIF socket provides an interface for the users to easily insert the IC chips they wish to test.

Lastly, the mobile device allows the user to select from a list of IC chips provided in ECE 385 and communicates the chip identifier with the control unit. The ESP32 microcontroller will act as an access point and produce an IP address. The user will merely need to connect to the ESP32’s Wi-Fi network and type the IP address in a web browser to access the application.

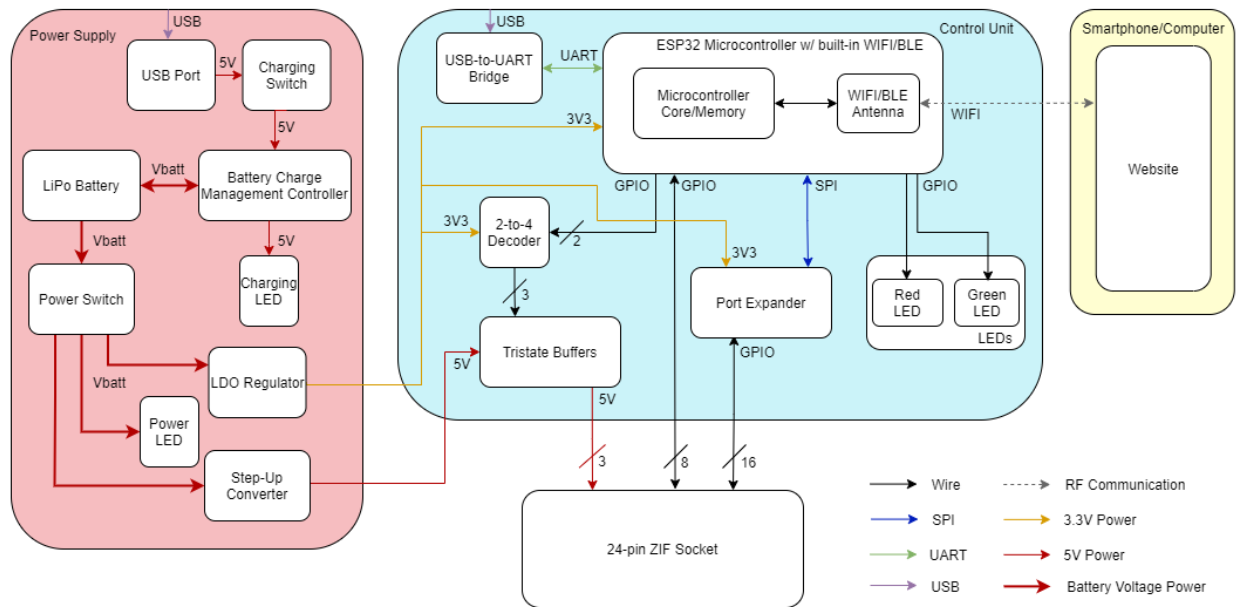


Figure 2: Automated IC Chip Tester Block Diagram

Comparing the final block diagram (Figure 2) with the one in our design document, we made some slight modifications to the power supply subsystem. Specifically, we decided to use one charge status LED instead of adding two more that would give the user an estimate on battery capacity. Because LiPo batteries should not be charged over 4.2 V or discharged under 3 V, our team wanted to alert the user if these conditions were met [4]. Moreover, LiPo batteries should not be stored fully charged because they might get damaged that way [5]. The primary reason we made the decision to omit the extra LEDs was because having one provides a much cleaner interface and we could inform the user of good charging habits in the product disclaimer. Additionally, while performing battery charging tests, we found that the battery did not rise above 3.8 V.

## 2 Design

### 2.1 Power Supply

Proper power distribution and control is vital for the project's full functionality and integration. During operation, the power supply provides 3.3 V to the ESP32 microcontroller, 2-to-4-line decoder, and the port expander. Additionally, it must also provide a constant 5 V to the tristate buffer. The subsystem also supports quick and easy battery charging capabilities through a USB Type B port and a dedicated charge management controller.

#### 2.1.1 Lithium Polymer Battery

Our team decided to use a 3.7 V single cell LiPo battery to power the device. Initially, we were considering using either 4 rechargeable AAA batteries or a wall socket to power our device. While these options would work in theory, they are not as user friendly. Specifically, the AAA batteries would need a separate plastic fixture to house them, thereby reducing the portability factor. The wall socket would also affect portability and require a converter to drop the wall's voltage to a safe level. As for the LiPo battery, its sleek design, minimal weight, and electrical specifications make it a great candidate for our project.

To ensure that the user has 3 hours of runtime, we calculated the minimum battery capacity to meet this requirement:

$$\text{Capacity [Ah]} = \text{Current [A]} * \text{Runtime [h]} \quad (1)$$

$$\text{Capacity [Ah]} = 0.564 \text{ [A]} * 3 \text{ [h]} = 1.692 \text{ [Ah]} \quad (2)$$

Based on the calculated minimum capacity of 1.692 Ah, our team chose a battery with 2.0 Ah of typical capacity to ensure our battery meets the high-level requirement (ESP32-WROOM: 500 mA, 2-to-4-line decoder: ~24 mA, ~18.5 for USB-to-UART bridge (this was not used in our final design), ~1 mA for port expander, and 20 mA for tristate buffer, the total output current necessary is 563.5 mA) [6].

#### 2.1.2 USB Port

Our team chose to use a USB 3.0 type B port to charge the battery. The USB is omnipresent and widely used in many different applications, therefore eliminating any unnecessary cable-finding hassle for the user. Moreover, a USB type B cable is already given to students to program their FPGA boards in ECE 385; this small design choice has big impact since users only need to have one cable for the entire lab kit. A standard USB 3.0 will provide 5 V and upwards of 1.5 A to charge the battery [7].

#### 2.1.3 Power Switch

The power switch controls the on/off state for the entire device. It protects the integrity of the device and preserves the battery life. Additionally, the power switch provides the user a convenient physical switch for quick on/off capability. The switch is rated to handle 650 mA of current and allows enough overhead to protect the integrity of the device [8].



### 2.1.4 Charging Switch

The charging switch controls the on/off state for charging. Adding a switch grants the user the added feature of leaving the USB always plugged in while keeping the switch in the off state. This added design feature eliminates the hassle of repeatedly plugging and unplugging the cable, especially during those long nights or all-nighters.

### 2.1.5 Power LED

A dedicated power LED indicates when the battery is powering the device. This added safety feature helps illustrate that the device is turned on.

### 2.1.6 Charging LED

When the battery is being charged through the USB port, an orange charging LED will turn on. This added feature tells the user when the battery is charging, and when the LED turns off, the battery is fully charged.

### 2.1.7 3.3 V Low-Dropout (LDO) Regulator

This voltage regulator circuit (see Figure 3) takes the battery voltage as an input and outputs a constant 3.3 V. The output of the regulator is sent to all other components on the printed circuit board, besides the tristate buffer. Furthermore, this regulator, the TPS77733, has a built-in enable pin that allows an open circuit if the input voltage drops too low [9]. The regulator will supply a maximum of 542 mA of current to the components that need 3.3 V. Therefore, to ensure enough overhead, our team chose a regulator that can handle up to 750 mA of current.

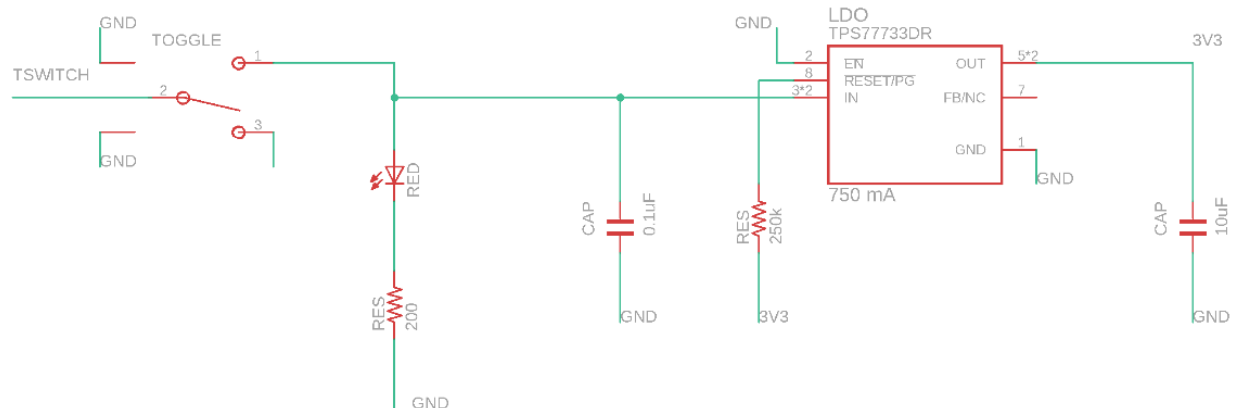
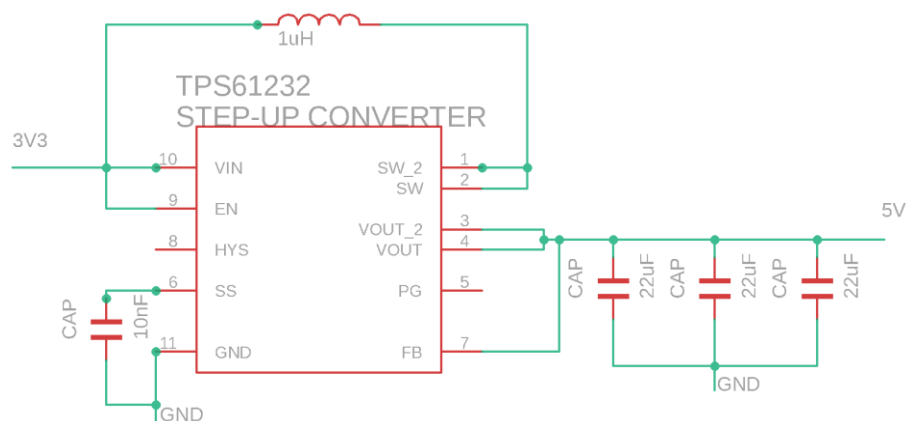


Figure 3: EAGLE Schematic of 3.3 V Regulator Circuit

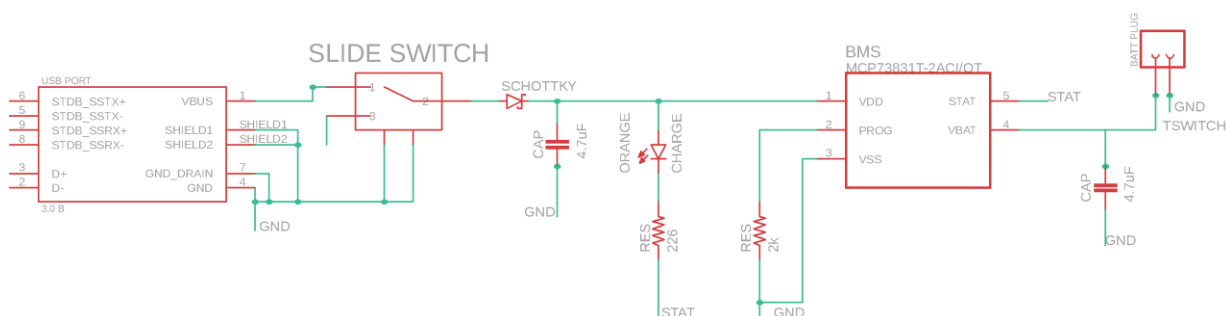
### 2.1.8 3.3 V/5 V Synchronous Boost Converter

This switch-mode boost regulator circuit (see Figure 4) takes the 3.3 V output from the LDO regulator and boosts it to 5 V ensure adequate operation of the tristate buffer. The reason we chose to include a 5 V voltage regulator alongside the 3.3 V one is to ensure that our device is compatible with the  $V_{CC}$ 's for the IC chips we are testing.



### 2.1.9 Charging Subsystem

The charging subsystem circuit (see Figure 5) is required to safely ensure that our LiPo battery is charged with a CC/CV methodology. We decided to use the MCP7381 Li-Polymer Charge Management Controller from Microchip for its constant-current/constant voltage charge algorithm with selectable preconditioning and charge termination. LiPo batteries are charged at a constant current until the battery reaches its nominal voltage. At this point, the current decays asymptotically to ensure that the battery is not overcharged. This is the crucial role of the BMS to ensure the longevity of our LiPo battery.



## 2.2 Control Unit

The control unit is responsible for properly handling the Wi-Fi signal from the web application as well as managing the digital signals sent to the chip under test. This subsystem consists of an ESP32 microcontroller, SPI-based port expander, 2-to-4 decoder, tristate buffers, and LEDs. Additionally, a USB-to-UART bridge is used to program the ESP32 microcontroller but is not a physical component included on the PCB since it is only necessary during development and should not be accessible by the user. Appendix C contains the Eagle schematic of this subsystem with all the connections made between each component and the interconnects to the power supply subsystem.

### 2.2.1 ESP32 Microcontroller

The ESP32 microcontroller serves as the centralized computational power of the project. Several design specifications were taken into account when determining which microcontroller to use. The first consideration was the operating voltage of the microcontroller and its compatibility with the operating conditions of the chips under test. The ESP32 family of microcontrollers operate at 3.3 V whereas the ATmega family of microcontrollers operate at 5 V and have a recommended input voltage of 7 V to 12 V [10]. All the IC chips provided in ECE 385 are CMOS chips meaning that they require 5 V of power, but only 2.0 V for inputs. ATmega microcontrollers provide the benefit of being able to directly power the IC chips through the General-Purpose Input Output (GPIO) pins without needing the voltage to be stepped up, but the additional voltage for digital input values is unnecessary. Since the project is being powered by a LiPo battery with a nominal voltage of 3.7 V, using an ATmega device would require the power to be stepped up or an additional battery wired in series to provide at least 7 V of input power. The ESP32 microcontroller does not provide the benefit of 5 V to power the IC chips under test but does operate above the 2.0 V threshold for logical high inputs and operates close to the nominal voltage of the LiPo battery, thereby minimizing loss in the LDO regulator.

Another major benefit of the ESP32 family of microcontrollers is that some of them are equipped with a built-in Wi-Fi/BLE antenna. Using the Wi-Fi capabilities within the ESP32, the web application can communicate with the dedicated server on the microcontroller so that users can select which chip they would like to test. The built-in Wi-Fi functionality of the ESP32 eliminates the need for external hardware to handle the communication as well as provides faster, more reliable communication since an external chip for Wi-Fi communication would need to transfer sent and received data to UART to pass it to the microcontroller. This adds more risk of failing hardware preventing the project from functioning properly. The ESP32-WROOM-32E model of microcontrollers provides the necessary voltage specifications for the IC chips under test and contains the built-in Wi-Fi capabilities required for our project.

### 2.2.2 Port Expander

The digital signals sent to the chips under test need to be wired in parallel with each other to read and write logic to and from the chips. This implies that at least 16 GPIO pins on the microcontroller must be dedicated to handling signals to and from the ZIF socket in order to handle the largest IC chip provided in the ECE 385 lab. However, the ESP32 microcontroller has a limited amount of GPIO pins preventing this since there are only eight available GPIO pins to control the test signals. Using a port expander provides 16 additional GPIO pins by sending 16 bits of information through either I2C or SPI protocol which get parallelized into 16 separate GPIO lines by the port expander.

The port expander acts as a 16-bit register with each index able to be individually accessed. Writing to the port expander can occur in 1-bit, 1-byte, or 2-byte intervals by sending serial information through I2C or SPI and each bit being placed into its respective index. Reading from the port expander can occur in the same intervals as a write operation but with the reverse process of serializing the parallel data. Connecting the port expander to the ESP32 microcontroller allowed our project to be able to handle up to 24-pin DIP chips by utilizing the sixteen GPIO on the port expander in conjunction with the eight available GPIO on the microcontroller.

An important requirement for our project is that it provides fast results to the user. The port expander is a bottle neck in the performance of our device since 16 signals need to be parallelized or serialized for every read or write operation. I2C communication is limited to a 400 kHz maximum frequency, whereas SPI protocol has a maximum frequency of 10 MHz [8]. By using an SPI-based port expander rather than I2C-based port expander, our project can execute an entire test suite for a chip under test nearly 25 times faster. This increased speed allows our project to meet the set latency constraint of two milliseconds.

### 2.2.3 Tristate Buffers

The IC chips under test require 5 V to power them in contrast to the 3.3 V used for logic signals. The  $V_{cc}$  pin has three possible locations on a DIP chip – top right pin, pin 4, or pin 5. To supply the correct pin with 5 V, the 3.3 V output from the ESP32 needs to be increased while also allowing for 3.3 V logic signals when the corresponding pin is not  $V_{cc}$ . Using tristate buffers with regulated 5 V from the power supply as input and control signals from the ESP32 as enable bits, the output from the tristate buffer will be 5 V for the pin corresponding to  $V_{cc}$ , and high impedance for the other two pins.

An alternate solution to this would be to use level shifters that step up the 3.3 V ESP32 signals to 5 V. However, this requires more circuitry and has higher noise than only using tristate buffers. Both provide the same output, but the level shifters would also require a regulated 3.3 V from the power supply and the output would have slightly more variance.

### 2.2.4 2-to-4 Decoder

Like the port expander, the purpose of the decoder is to solve the issue of limited GPIO pins on the ESP32 microcontroller. The enable pins of the tristate buffers are required to have only one active at a time but require three separate signals to differentiate between each enable pin. By utilizing a 2-to-4 decoder to control which enable pin is active on the tristate buffers, only two GPIO pins are required from the ESP32 rather than three without the use of the decoder. Additionally, the enable pins for the tristate buffers are inverted and the outputs from the decoder are inverted. Since these match, this negates the need to invert the control signals in the firmware of the ESP32.

### 2.2.5 LEDs

The purpose of the green and blue LEDs is to provide the user with a quick visual of whether the chip is working as expected or not. If the tested IC chip is functioning properly, therefore passing all test vectors, the green LED will turn on and stay on until the user tests another chip. Else, if the chip is faulty, the blue LED will turn on as soon as a single test vector is failed.

### 2.2.6 USB-to-UART Bridge

The USB-to-UART connection is required to upload firmware to the ESP32 during development. This component was originally designed to be included on the PCB and connected to the on-board USB port, however when ordering parts, the specific part was out of stock. Rather than choose a replacement part to take its spot, we opted to purchase a USB-to-UART bridge with an attached USB male port. The component instead connected to the PCB through female headers for the UART signals and was plugged directly into a computer. This ended up being a good design decision because the USB port connected to the power supply subsystem did not work as intended. The added benefit of the separated USB-to-UART

bridge is that it prevents the user from accidentally or maliciously resetting the microcontroller firmware. Since the project has a Wi-Fi component, one of the ethical concerns of our project is preventing the user's exposure to cyber-attacks. If the USB-to-UART bridge had been connected to the on-board USB port, the microcontroller firmware could be easily reset by a third party with malicious intent and attack any user that connects to the device. The separated bridge prevents this from occurring if the third party does not have background knowledge of the PCB layout.

## 2.3 Wi-Fi Application

Along with the physical device, the Wi-Fi application is the other component of our project's user interface. The ESP32 generates an IP address that serves as the host for our web application.

This web application was built with an Arduino IDE and utilizing C++, React (JavaScript), HTML, and CSS programming languages. We chose to use the ESP32's Wi-Fi module rather than the BLE module because the latter would require Arduino app development, which we had limited experience with.

Originally, we were planning to utilize the ESP32's Serial Peripheral Interface Flash File System (SPIFFS) to host extra files, excluding the C++ file we needed to program the firmware. However, we started facing problems with data storage, especially when importing the various images of the chips. We then switched over to a React.js application. This provided a preset framework for React visual tools often used in industry and less memory on the ESP32's flash.

The Wi-Fi application has two interfaces for the user: a home and a results screen. Figure 4A shows the home screen for our application where the user can select the chip they wish to test. Figure 4B depicts the results screen which appears after the user clicks the "Submit" button. A more in-depth discussion of the testing algorithm is addressed in Section 2.3.1.

For the backend of our application, we utilized HTTP API requests through the aWOT library. This library is Arduino-based and is compatible with various board architectures. HTTP API requests use a client/server communication and has all possible CRUD (create, retrieve, update, delete) operations.

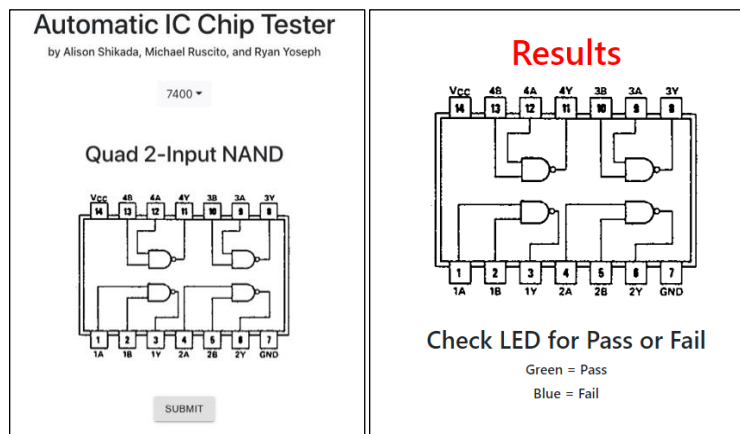


Figure 6: A (Left) Home Screen UI. B (Right): Result Screen UI

### 2.3.1 Flowchart

Figure 6 shows a flowchart of our algorithm for testing IC chips. In order to test an IC chip, users only need to click on the dropdown menu and select the appropriate chip number. Once selected, our application will update with the appropriate chip title and pin layout image. After confirming that this chip is correct, users will need to click the “Submit” button at the bottom of the page to run the preset test vectors on the firmware. The results screen that will appear once the testing is completed. The blue and green LEDs on the device will tell users whether or not their current IC chip is operational.

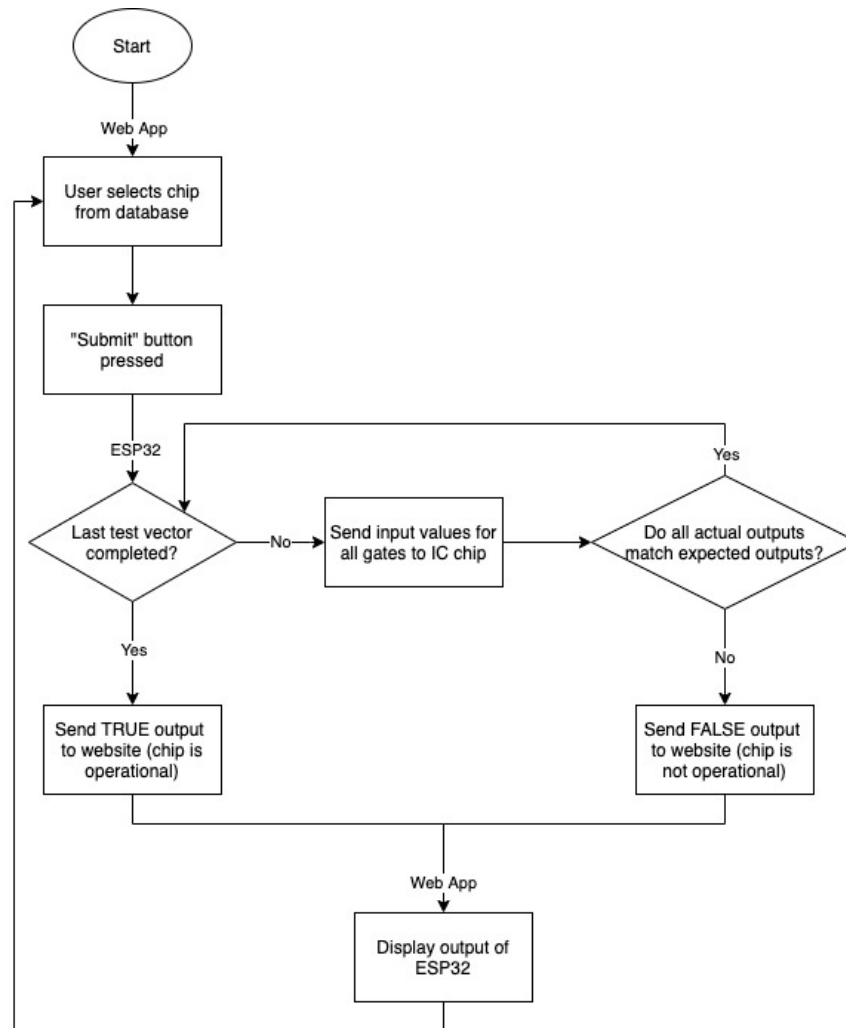


Figure 7: Algorithm Flowchart

### 3. Design Verification

The complete requirement and verification table for all subsystems is included in Appendix A.

#### 3.1 Power Supply

To ensure proper functionality of our project, our team confirmed that both voltage regulators were able to maintain their designed constant voltage values. Specifically, the 3.3 V LDO regulator must be capable of outputting  $3.3\text{ V} \pm 0.3\text{ V}$  at 600 mA and the 5 V voltage regulator must provide  $5\text{ V} \pm 0.5\text{ V}$  and be able to handle 20 mA sourcing to the tristate buffers. To test these requirements, a multimeter and power supply were used. Once we completed soldering, we began by sending a constant 3.7 V input to the VBAT test pad and GND test pad. We probed the test pads labeled 3.3 V and 5 V and wrote down their readings as 3.295 and 5.10 V respectively. The current on the power supply also read 70 mA. This is because the ESP32 was not yet programmed so it was not sinking a substantial amount of current. However, once the design was complete and functioning to its full capacity, the team noticed that there were no issues with power and that by probing the test pads once again, the values were 3.295 V and 5 V. Therefore, the power supply outputs were validated.

#### 3.2 Control Unit

The requirements for the control unit subsystem are split into 2 sections – peripheral hardware and ESP32 microcontroller. This is because the ESP32 is such a central part of the project. Also, passing the requirements related to the microcontroller are heavily dependent on its firmware development, whereas the peripheral hardware requirements are more reliant on the hardware.

##### 3.2.1 Peripheral Hardware

The control unit's peripheral hardware has two requirements, both related to the voltage readings of specific pins on the ZIF socket. The first requirement is that the port expander must be able to parallelize 16-bit outputs from the SPI communication with the ESP32. This ensures that the port expander has an operational write functionality. The verification method for testing this consisted of uploading firmware to the ESP32 that writes logical low values to select ZIF socket pins, and logic high values to other pins. The pins are then probed with a voltmeter to confirm that the values are correct. The tolerance of the logic values for the IC chips is that a logical low must be less than 0.8 V, while a logical high is greater than 2.0 V. However, since the ESP32 operates at 3.3 V, we should expect to read logical highs close to 3.3 V. The results of this verification were successful as shown by the results of our tests in Table 1.

Table 1: Verification Results of Port Expander Write Operations

Logic Value	Measured Voltage
Logical Low	0.000 V
Logical High	3.285 V

The second requirement for the peripheral hardware is that 5 V is properly supplied to the  $V_{cc}$  pin of the chip under test. The 2-bit signal from the ESP32 must be decoded to the tristate buffers which output 5 V to the selected pin and high impedance to other two pins. Similar to the previous requirement, the verification method consists of uploading firmware to the ESP32 that selects a  $V_{cc}$  pin and measuring the

voltage out of the ZIF socket pins with a voltmeter. The voltage of the selected pin must be  $5\text{ V} \pm 0.7\text{ V}$ . The other two pins must be floating as to not interfere with the digital logic values sent by the ESP32 during testing. Table 2 shows the successful results of this verification.

Table 2: Verification Results of  $V_{cc}$  Pin Selection

Selected Pin	Measured Voltage
$V_{cc}$	5.020 V
Other Pins	Floating

### 3.2.2 ESP32 Microcontroller

The requirements for the ESP32 heavily rely on successful firmware development in conjunction with the rest of the control unit hardware. The first requirement is that the microcontroller's firmware must contain a library of test suites for all 18 chips provided in the ECE 385 lab. For each of the chips, the firmware contains a function that executes the entire test suite when the chip is selected by the user on the Wi-Fi application. Figure 8 shows the process that is taken in each of these functions for the respective chips. First the  $V_{cc}$  and GND pins are determined, followed by declaring each pin as either an input or an output. Then, test vectors are derived from the chip's truth tables and executed.

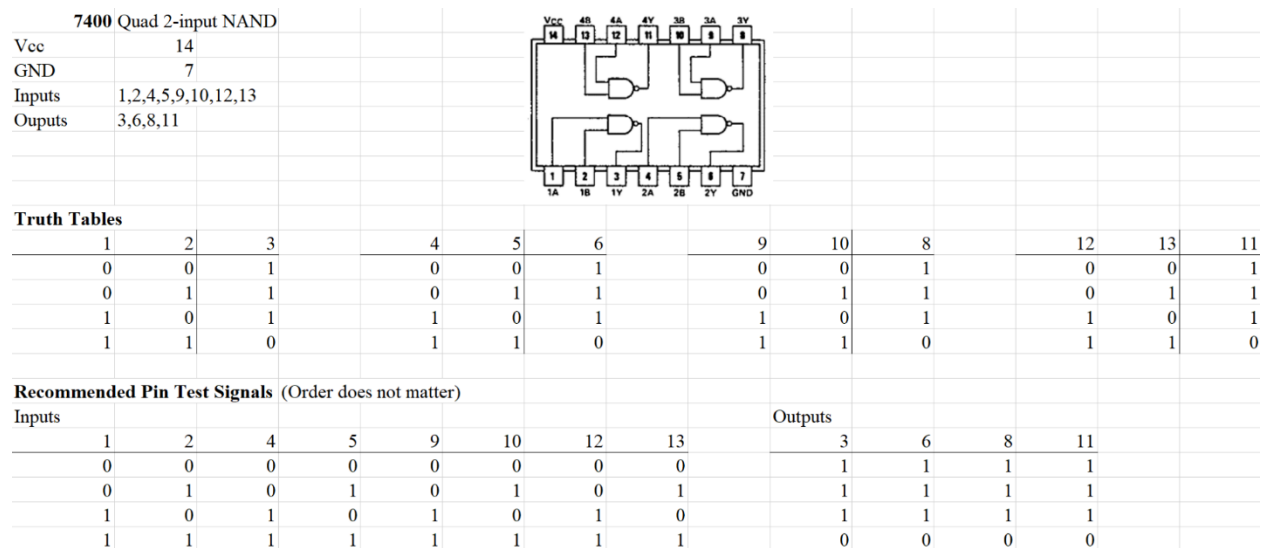


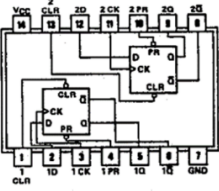
Figure 8: Test Suite for 7400 Chip

Figure 9 shows the same steps being followed for sequential logic chips as well, with the key differences being that function tables are used rather than truth tables, and some test vectors have expected outputs of "X". When an output is labeled with an "X", this means that the firmware does not read the outputs until a second write operation is performed. This is because the input values are set with the clock low, then the clock is triggered high, and the outputs are measured.



<b>7474 Dual D Positive Edge Triggered Flip Flop</b>													
Vcc	14												
GND	7												
Inputs	1,2,3,4,10,11,12,13												
Outputs	5,6,8,9												
Clock	3,11												



<b>Function Tables</b>													
4 (Pre)	1 (CLR)	3 (CLK)	2 (D)	5 (Q)	6 (Q')	10 (Pre)	13 (CLR)	11 (CLK)	12 (D)	9 (Q)	8 (Q')		
0	1	X	X	1	0	0	1	X	X	1	0		
1	0	X	X	0	1	1	0	X	X	0	1		
1	1	↑	1	1	0	1	1	↑	1	1	0		
1	1	↑	0	0	1	1	1	↑	0	0	1		
1	1	0	X	Q	Q'	1	1	0	X	Q	Q'		

<b>Recommended Pin Test Signals (Order matters)</b>													
Inputs									Outputs				
1	2	3	4	10	11	12	13		5	6	8	9	Test
1	0	0	0	0	0	0	1		1	0	0	1	Preset
0	0	0	1	1	0	0	0		0	1	1	0	Clear
1	1	0	1	1	0	1	1		X	X	X	X	Load 1
1	1	1	1	1	1	1	1		1	0	0	1	
1	0	0	1	1	0	0	1		X	X	X	X	Load 0
1	0	1	1	1	1	0	1		0	1	1	0	
1	1	0	1	1	0	1	1		0	1	1	0	Hold

Figure 9: Test Suite for 7474 Chip

Verification of this requirement is done in conjunction with the second requirement which is that the ESP32 must be able to read from all 24 of the ZIF socket pins with low logic states in the range -0.3 V to 0.8 V and high logic states in the range 2.47 V to 3.6 V indicating that the port expander has an operational read functionality. Verifying these requirements consists of testing a chip and reading the serial monitor outputs compared to the expected outputs. Figure 10A, B, and C show the steps of this process. Figure 10A shows the first test vector in the 7400 test suite, but the bits are not arranged in the order that they appear on the ZIF socket. Figure 10B is the rearrangement of these bits into the expected output from the test vector, and Figure 10C is the serial monitor outputs from the test suite. The hex value outlined in Figure 10C matches the 16-bit value of Figure 10B demonstrating the success of both requirements.

A. Inputs									Outputs				
1	2	4	5	9	10	12	13		3	6	8	11	
0	0	0	0	0	0	0	0		1	1	1	1	
0	1	0	1	0	1	0	1		1	1	1	1	
1	0	1	0	1	0	1	0		1	1	1	1	
1	1	1	1	1	1	1	1		0	0	0	0	

B. 0010 0100 0010 0100

C. Testing 7400 Chip

2424

2D36

362D

1B1B

Result:1

Test Run Time: 809 microseconds

Figure 10: A: 7400 Chip Test Vectors. B: Reorganized Test Vector Bit Values. C: Serial Monitor Output of 7400 Chip Test

The final requirement of the ESP32 is that the device's internal logic must be able to determine if a chip is working properly in under 2 ms. This latency constraint is to ensure that the user is provided with fast results. This requirement is verified using the `micros()` function in the firmware of the ESP32 microcontroller to calculate the time difference between when the system receives data from the user specifying which chip to test and the time the chip is determined to be working or not. The `micros()` function returns the number of microseconds that have passed, so test suites must run in under 2000  $\mu$ s. Figure 11 shows the serial monitor output determining the total run time of a test suite. On the serial monitor, result "1" indicates that the chip works and result "0" indicates that the chip is broken. When a chip fails a test vector, the test suite is interrupted and immediately informs the user of the broken chip. As a result, a failing chip will not execute the entire test suite and will therefore have a faster run time. Figure 12 graphs the total run time for all the test suites in  $\mu$ s. The dashed red line indicates the 2 ms latency constraint and demonstrates the success of this requirement since all chips are beneath it.

Testing 7474 Chip Result:1 Test Run Time: 1139 microseconds	Testing 7474 Chip Result:0 Test Run Time: 663 microseconds
---	--

Figure 11: Serial Monitor Output from Passing 7474 Test Suite (Left) and Failing 7474 Test Suite (Right)

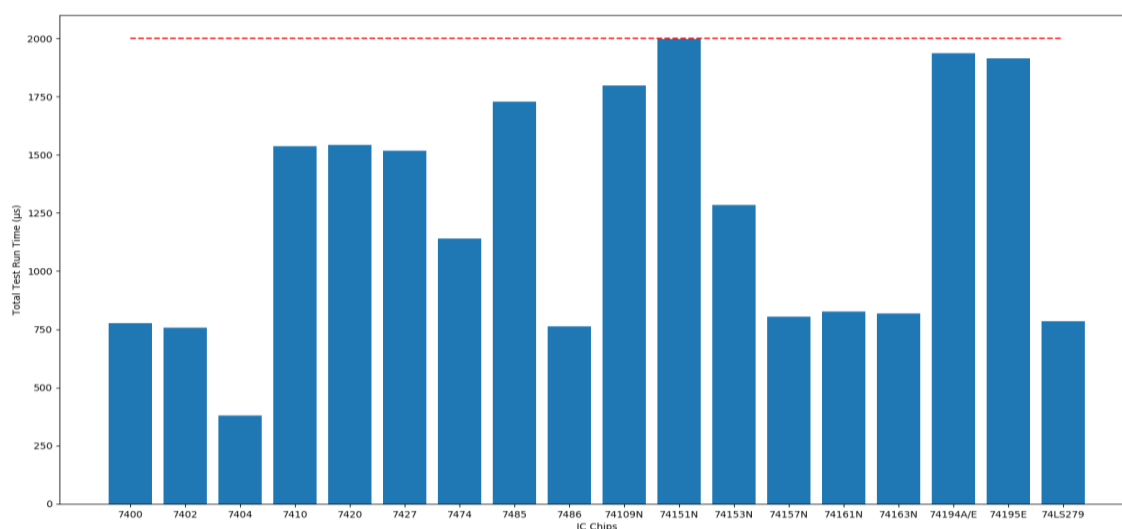


Figure 12: Total Run Time for Each Passing Test Suite in Microseconds

### 3.3 Wi-Fi Application

To verify that our results are in accordance with the items listed in Appendix A, we applied various tests to our UI. We were able to connect both a laptop and a smartphone to our web application once the devices were on the ESP32's network. This connection was stable and reliable because neither the network nor the application glitched or buffered while testing a series of IC chips. The dropdown menu on the home page does list all IC chips offered in ECE 385. Upon testing a given IC chip, the output of that chip is consistent and delivers the correct output to the LEDs. Lastly, we are able to test different IC chips sequentially without restarting the server or reconnecting to the IP address. Both the firmware and the application UI contribute to the sequential IC chip testing.

## 4. Costs and Schedule

### 4.1 Parts

Table 3: Parts Cost

Part	Manufacturer	Retail Cost (\$)	Quantity	Total Cost (\$)
UMFT234XD-WE	FTDI	12.53	1	12.53
CD74HCT125E	Texas Instruments	.77	2	1.54
ESP32-WROOM-32E	Espressif Systems	2.80	2	5.60
SN74LVLC1G139DCUT	Texas Instruments	.89	2	1.78
MCP23S17T-E/SO	Microchip Technology	1.32	2	2.64
TPS77733DR	Texas Instruments	3.06	2	6.12
MCP73831T-2ACI/OT	Microchip Technology	.59	2	1.18
TPS61230DRCT	Texas Instruments	2.71	2	5.42
455-1165-ND	JST Sales America Inc.	.10	2	0.20
GSB3211311WEU-ND	Amphenol ICC	2.31	2	4.62
732-4984-6-ND	Würth Elektronik	.18	4	0.72
516-1444-1-ND	Broadcom Limited	1.05	4	4.20
1497-1393-1-ND	SunLED	.41	2	0.82
1497-1310-1-ND	SunLED	.61	2	1.22
EG2362-ND	E-Switch	2.70	2	5.40
EG2483-ND	E-Switch	4.21	2	8.42
399-4950-1-ND	KEMET	.41	4	1.64
399-11948-1-ND	KEMET	.549	10	5.49
1276-1010-1-ND	Samsung Electro-Mechanics	.1	2	0.2
VS-MBRA140TRPBFCT-ND	Vishay General Semiconductor - Diodes Division	.59	2	1.18
P226AACT-ND	Panasonic Electronic Components	.24	2	0.48
RMCF2512FT10KOCT-ND	Stackpole Electronics Inc	.37	2	0.74
EG2526CT-ND	E-Switch	.36	2	0.72
399-15351-1-ND	KEMET	.16	4	0.64
399-7412-1-ND	KEMET	.50	2	1.00
HVCB2512FDC250KCT-ND	Stackpole Electronics Inc	6.58	2	13.16
296-37990-1-ND	Texas Instruments	3.01	2	6.02
490-4339-1-ND	Murata Electronics	.35	2	0.70
811-3619-1-ND	Murata Power Solutions Inc.	1.69	2	3.38
24-6554-10	Aries Electronics	11.15	2	22.15
CP2104-F03-GMR	Silicon Labs	1.84	1	1.84
LP605060JU + PCM + WIRES 70MM	Jauch Quartz	16.89	1	16.89
MMBT2222	Rochester Electronics, LLC	.79	2	1.58
AP2111H-3.3TRG1	Diodes Incorporated	.49	1	0.49
MAX1725EUK+T	Maxim Integrated	2.00	1	2.00
			<b>Total Parts Cost:</b>	<b>\$142.71</b>

## 4.2 Labor

Table 4: Labor Costs

Member	Hourly Wage	Hours per Week	Number of Weeks	Multiplier	Total
Michael Ruscito	\$40	15	16	2.5	\$24,000
Alison Shikada	\$40	15	16	2.5	\$24,000
Ryan Yoseph	\$40	15	16	2.5	\$24,000
				<b>Total Labor Cost:</b>	<b>\$72,000</b>

## 4.3 Total Costs

Table 5: Total Costs

Parts Costs:	\$142.71
Labor Costs:	\$72,000
<b>Total Costs:</b>	<b>\$72,142.71</b>

## 4.4 Schedule

Table 6: Schedule

Week	Alison Shikada	Michael Ruscito	Ryan Yoseph
3/1	<ul style="list-style-type: none"> <li>ESP32 Wi-Fi app or in house app</li> <li>RV block</li> </ul>	<ul style="list-style-type: none"> <li>Control unit design</li> <li>part numbers</li> <li>RV block</li> </ul>	<ul style="list-style-type: none"> <li>Power supply</li> <li>RV block</li> </ul>
3/8	<ul style="list-style-type: none"> <li>Revise Design based on feedback</li> </ul>	<ul style="list-style-type: none"> <li>Revise Design based on feedback</li> </ul>	<ul style="list-style-type: none"> <li>Revise Design based on feedback</li> </ul>
3/15	<ul style="list-style-type: none"> <li>Order parts</li> </ul>	<ul style="list-style-type: none"> <li>Order parts</li> </ul>	<ul style="list-style-type: none"> <li>Order parts</li> </ul>
3/22	<ul style="list-style-type: none"> <li>Set up simple ESP32 server</li> </ul>	<ul style="list-style-type: none"> <li>Begin development on ESP32 firmware for test suites</li> </ul>	<ul style="list-style-type: none"> <li>Begin development on ESP32 firmware for MCP7381 communication</li> </ul>
3/29	<ul style="list-style-type: none"> <li>Test simple CMOS gate with ESP32 algorithm</li> </ul>	<ul style="list-style-type: none"> <li>Solder components to PCB</li> </ul>	<ul style="list-style-type: none"> <li>Solder components to PCB</li> </ul>
4/5	<ul style="list-style-type: none"> <li>Implement 2-way communication with HTTP API</li> <li>Create layouts on web application for 9 out of 18 chips</li> <li>Draft algorithm for testing chip deficiency</li> </ul>	<ul style="list-style-type: none"> <li>Create library of test vectors for all 18 IC chips given in ECE385, specifically designating which chips are sensitive to order of signals</li> <li>Solder components on PCB</li> </ul>	<ul style="list-style-type: none"> <li>Solder components on PCB and use testpads to measure board output.</li> <li>Place order for PCB in the third round</li> </ul>

Week	Alison Shikada	Michael Ruscito	Ryan Yoseph
4/12	<ul style="list-style-type: none"> <li>Depending on PCB completion, fully integrate 1 CMOS chip</li> <li>If not, integrate a mock chip for testing vectors</li> <li>Full integration of 18 chips on web app</li> </ul>	<ul style="list-style-type: none"> <li>Develop any test firmware required for hardware debugging purposes</li> <li>Debug any hardware issues for control unit</li> <li>Aid in programming test vectors</li> </ul>	<ul style="list-style-type: none"> <li>Begin firmware development and research on Saturday, April 10th.</li> </ul>
4/19	<ul style="list-style-type: none"> <li>Complete full integration of test vectors with PCB</li> <li>Modify website UI if needed</li> </ul>	<ul style="list-style-type: none"> <li>Test/debug device performance</li> </ul>	<ul style="list-style-type: none"> <li>Test/debug device performance</li> </ul>
4/26	<ul style="list-style-type: none"> <li>Implement changes found in Mock demo</li> <li>Deliver demo</li> <li>Draft Final Paper</li> </ul>	<ul style="list-style-type: none"> <li>Deliver demo</li> <li>Draft Final Paper</li> </ul>	<ul style="list-style-type: none"> <li>Deliver demo</li> <li>Test/debug device performance</li> <li>Draft Final Paper</li> </ul>
5/3	<ul style="list-style-type: none"> <li>Deliver Presentation</li> <li>Finish final paper</li> </ul>	<ul style="list-style-type: none"> <li>Deliver Presentation</li> <li>Finish final paper</li> </ul>	<ul style="list-style-type: none"> <li>Deliver Presentation</li> <li>Finish final paper</li> </ul>

## 5. Conclusion

### 5.1 Accomplishments

The Automated IC Chip Tester can successfully determine whether a user selected IC chip is operational or not. Through the use of our Wi-Fi application, the user is provided with a clean, intuitive UI for easy testing with a full testing suite. With the inclusion of the LiPo battery, this device provides a compact, portable, and rechargeable solution suited for students to unit test their IC chips in ECE 385.

### 5.2 Uncertainties

There were a few design flaws in our project that a second prototype would ideally address. The first being that the battery charging system is not integrated on the same PCB as the automated IC chip tester. This issue is not severe because the battery is easily removable from the device and can be charged separately, however for ease of use for the user, charging should be integrated on the same device. The reason for this issue is due to a faulty PCB layout in which the pins of the BMS chip were not labelled properly shorting the chip as a result. If we had an additional round of PCB orders, we would have been able to solve this issue.

The other shortcoming in our project is that the website does not update with the test results. Instead, we had to rely on the LEDs on the PCB to indicate whether a chip was working. This is because the HTTP request for the results is too fast for the ESP32 to update it in time. So, the solution to this is to implement a web socket instead which would allow the user to check the UI for the updated results rather than relying on the LEDs.

### 5.3 Ethical considerations

In electing to design a Web application as opposed to a BLE application, we have a higher exposure to cyber-attacks of malicious intent. IEEE's Code of Ethics, Section I, Policy 1 [11] plays a role in our attempt to protect the user from a breach of privacy. While we believe the benefits of a Wi-Fi application outweigh that of a BLE alternative, it is our responsibility to not compromise the security of our users. Additionally, it is our responsibility to not abuse the trust of our users by not extracting any data from their device.

Our solution to this security risk is to utilize the ESP32's soft access point (softAP) module. Usually, a router serves as an access point while a mobile device is a station. In our case, the ESP32 acts as the access point and generates its own Wi-Fi network. This softAP adds another layer of protection because a malicious user must first connect to the appropriate Wi-Fi network before accessing the application.

Otherwise, our project has several potential safety hazards. Batteries can be dangerous when used outside of the recommended operating conditions. If a battery is brought to extreme temperatures, it can become a fire hazard [12]. We address this issue by only using our project within environments of -18°C to 55°C. Using tristate buffers to control the battery drain, our project will also ensure that the power drawn from the batteries is within the safe operating conditions.

## 5.4 Future work

We have a few ideas for improving our project. We will address improvements first to our current model, and then to extrapolate our project to a larger scale.

First, for a more durable device, we would like to 3D print a case to protect the PCB and battery of our tester. With a more durable exterior, students may carry testers in backpacks without worrying about damaging the device. We would also like to interface our Wi-Fi application with the University of Illinois' Shibboleth system. This would provide a 2-Factor Authentication already recognized by the university as a secure solution to students' and faculty's smart devices.

If we were to expand our project's functionality, the first change we would like to make would be to create a larger tester to be able to handle larger sized chips. Currently, we are limited to a 24-pin ZIF socket due to the ESP32's limited GPIO pins. With more ESP32's and more hardware components, we could potentially handle those larger IC chips. Lastly, we would like to create a circuit tester as an added feature to our tester. If we could designate a certain number of pins to unit test a circuit at various points in its wiring, we could compare the expected inputs and outputs with that of a user-defined test. At this point, our device can determine if the input and output of a circuit is the same as an expected value. However, we would like to test designated "checkpoints" of a circuit rather than just the input and output.

## References

- [1] Y. H. Ng, Y. H. Low and S. Demidenko, "Improving Efficiency of IC Burn-In Testing," 2008 IEEE Instrumentation and Measurement Technology Conference, Victoria, BC, Canada, 2008, pp. 1685-1689. [Accessed March 1, 2021].
- [2] K. N. Tu, "Reliability challenges in 3D IC packaging technology," *Microelectronics Reliability*, vol. 51, no. 3, pp. 517–523, Sep. 2010.
- [3] Ronen, M. and Eliahu, M. (2000), "Simulation — a bridge between theory and reality: the case of electric circuits," *Journal of Computer Assisted Learning*. [Abstract]. Available: Wiley Online Library, <https://doi.org/10.1046/j.1365-2729.2000.00112.x>. [Accessed March 1, 2021].
- [4] *Maxamps.com Award Winning Batteries*. <https://www.maxamps.com/lipo-care.php> (accessed May 05, 2021).
- [5] DNK Power Company, "Complete Guide for Lithium Polymer (Lipo) Battery: History, Charging, Safety, Storage, and Care," Lithium ion Battery Manufacturer and Supplier in China-DNK Power, March 11, 2019. [Online]. Available: <https://www.dnkpower.com/lithium-polymer-battery-guide/>. [Accessed: March 5, 2021].
- [6] Texas Instruments, "High-Speed CMOS Logic Quad Buffer, Three-State", 54HC125 datasheet, Nov. 1997 [Revised Aug. 2003].
- [7] N. Stokes, "How to Charge Your Phone Faster," *techlicious.com*, Jul. 2, 2019. [Online]. Available: <https://www.techlicious.com/tip/charge-your-iphone-android-phone-faster/>. [Accessed: March 28, 2021].
- [8] E-Switch, "Series 100 Switches Toggle Switches – Miniature", 100SP1T2B4M6QE datasheet (accessed May 05, 2021).
- [9] Texas Instruments, "TPS77825 FAST-TRANSIENT-RESPONSE 750-mA LOW-DROPOUT LINEAR REGULATORS", TPS777 datasheet, Sept. 1999 [Revised Jan. 2004] (accessed Apr. 07, 2021).
- [10] Espressif Systems, "ESP32 Series Datasheet," v3.5, Aug. 2016 [Revised Jan. 2021].
- [11] IEEE, "IEEE Code of Ethics," *IEEE*. [Online] Available: <https://www.ieee.org/about/corporate/governance/p7-8.html>. [Accessed: February 18, 2021].
- [12] Energizer Brands, LLC., *Alkaline Manganese Dioxide Handbook and Application Manual*, Energizer, 2018.



## Appendix A

## Requirement and Verification Table

Table 6: Complete RV Table with Demo Point Allocation

Subsystem	Requirements	Verification	Points
Power Supply	<ul style="list-style-type: none"> <li>The 3.3 V LDO regulator must be capable of outputting <math>3.3\text{ V} \pm 0.3\text{ V}</math> at 600 mA. 8 points</li> <li>The 5 V voltage regulator must provide <math>5\text{ V} \pm 0.5\text{ V}</math> and be able to handle 20 mA sourcing to the tristate buffers. 7 points</li> </ul>	<ul style="list-style-type: none"> <li>Measure the output voltage of the LDO regulator by probing the 3.3 V and GND pads with a voltmeter to ensure the regulator outputs <math>3.3\text{ V} \pm 0.3\text{ V}</math>.</li> <li>Measure the output voltage by probing the 5 V and GND pads with a voltmeter to ensure the regulator outputs <math>5\text{ V} \pm 0.5\text{ V}</math>.</li> </ul>	15
Control Unit (Excluding ESP32)	<ul style="list-style-type: none"> <li>The port expander must be able to parallelize 16-bit outputs from the SPI communication with the ESP32. 3 points</li> <li>The 2-bit signal from the ESP32 must be decoded to the tristate buffers which output 5 V to the selected pin and high impedance to other 2 pins. 4 points</li> </ul>	<ul style="list-style-type: none"> <li>Firmware housed on the ESP32 writes each of the 16 bits in the SPI communication. Verify that the corresponding output pin on the ZIF socket matches with a multimeter (<math>&lt;0.8\text{ V} = \text{Low}</math>, <math>&gt;2.0\text{ V} = \text{High}</math>).</li> <li>Measure the voltage out of the ZIF socket pin in the top right, compared to the voltage of pins 4 and 5. The voltage of the selected pin must be <math>5\text{ V} \pm 0.7\text{ V}</math>. The other 2 pins must match the ESP32's output within the tolerances described for its logic values.</li> </ul>	7
ESP32 Microcontroller	<ul style="list-style-type: none"> <li>The ESP32 must be able to read from all 24 of the ZIF socket pins with Low logic states in the range <math>-0.3\text{ V}</math> to <math>0.8\text{ V}</math> and High logic states in the range <math>2.47\text{ V}</math> to <math>3.6\text{ V}</math>. 3 points</li> <li>The device's internal logic must be able to determine if a chip is working properly in under 2 milliseconds. 2 points</li> </ul>	<ul style="list-style-type: none"> <li>Probe each pin of the ZIF socket with <math>3.3\text{ V}</math> and compare the ESP32's input reading from the console. Repeat the process but probing with <math>0\text{ V}</math>.</li> <li>Use <code>micros()</code> function in the firmware of the ESP32 microcontroller to calculate the time difference between when the system receives data from the user specifying which chip to test and the time the chip is determined to be working or not.</li> </ul>	10

	<ul style="list-style-type: none"> <li>• The microcontroller's firmware must contain a library of test suites for all 18 chips provided in the ECE 385 lab. 5 points</li> </ul>	<ul style="list-style-type: none"> <li>• Test chips through the user interface using both chips that have been manually tested and verified to be working as well as chips that are broken. Compare the device's diagnostic of the chips to the known state.</li> </ul>	
Wi-Fi Application	<ul style="list-style-type: none"> <li>• Web application and wireless network generated from ESP32 using Soft Access Point and application is accessible from multiple devices 5 points</li> <li>• Web application allows for chip selection of each of the 18 IC chips provided in ECE 385 lab kit. 5 points</li> <li>• The Wi-Fi app can display the correct testing results for the chip. 3 points</li> <li>• Wi-Fi app generates input and receives output from ESP32 for custom unit tests with a minimum of 95% accuracy 3 points</li> <li>• Wi-Fi app can execute a second test following the completing on a first test without needing to reprogram the ESP32 or reconnect to the network 2 points</li> </ul>	<ul style="list-style-type: none"> <li>• Connect to the web app via a smartphone and confirm that the webpage is responsive to input. Repeat the process by accessing it via a computer or other device.</li> <li>• Using the drop-down menu feature of the web application, choose each of the chips and verify that the page updates for the selected chip.</li> <li>• The web application correctly displays the same output as determined by the ESP32 microcontroller and corresponding LEDs.</li> <li>• Send 100 inputs on the Wi-Fi app to the ESP32 with a working IC chip (previously tested) in the ZIF socket and check that the returned outputs match the expected logic values.</li> <li>• Load a chip into the ZIF socket and run the corresponding test for it. Then, without restarting the system, swap the chip for a new one and run the new corresponding test suite.</li> </ul>	18
<b>Total Points</b>			50

## Appendix B PCB Layout

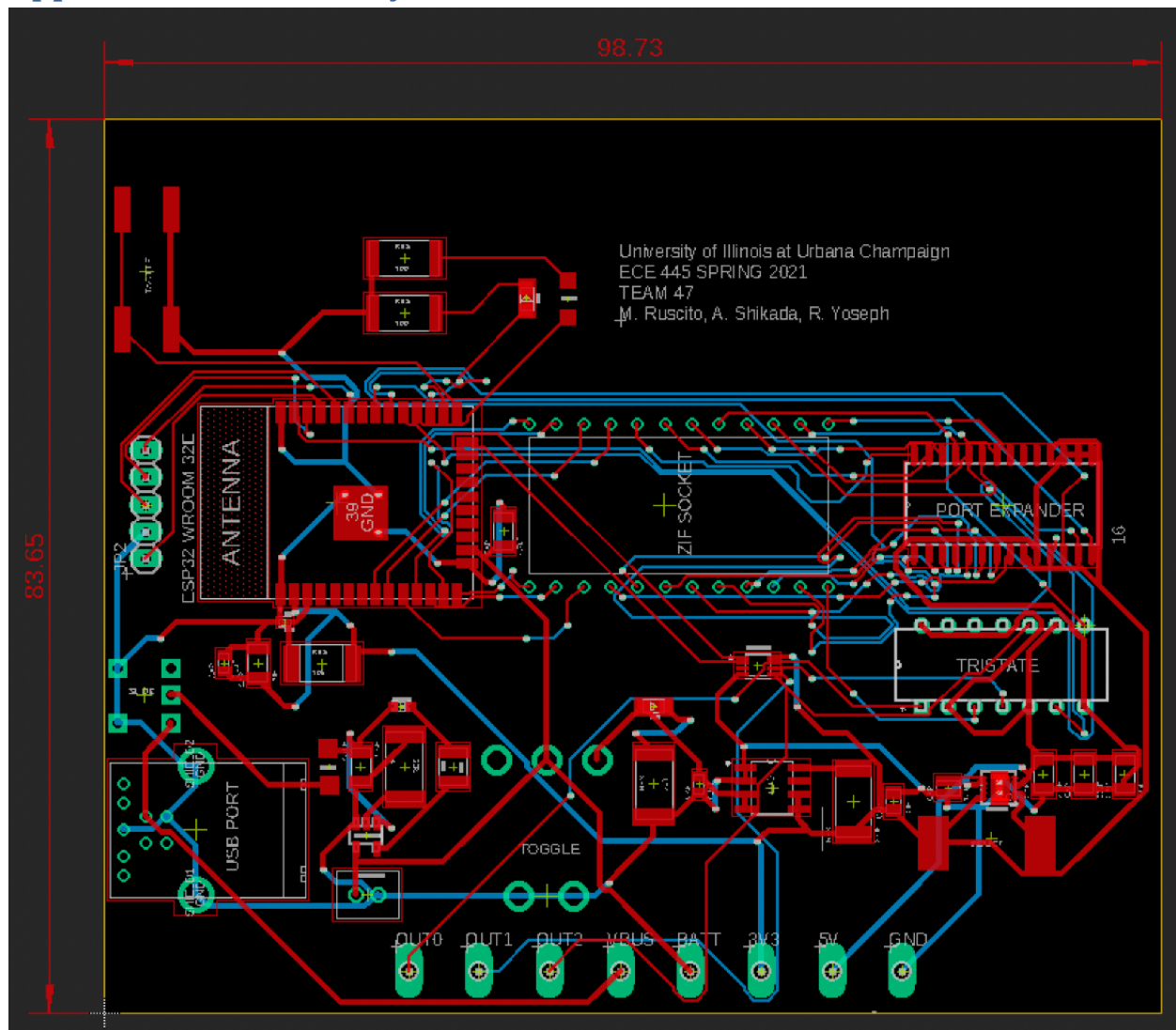
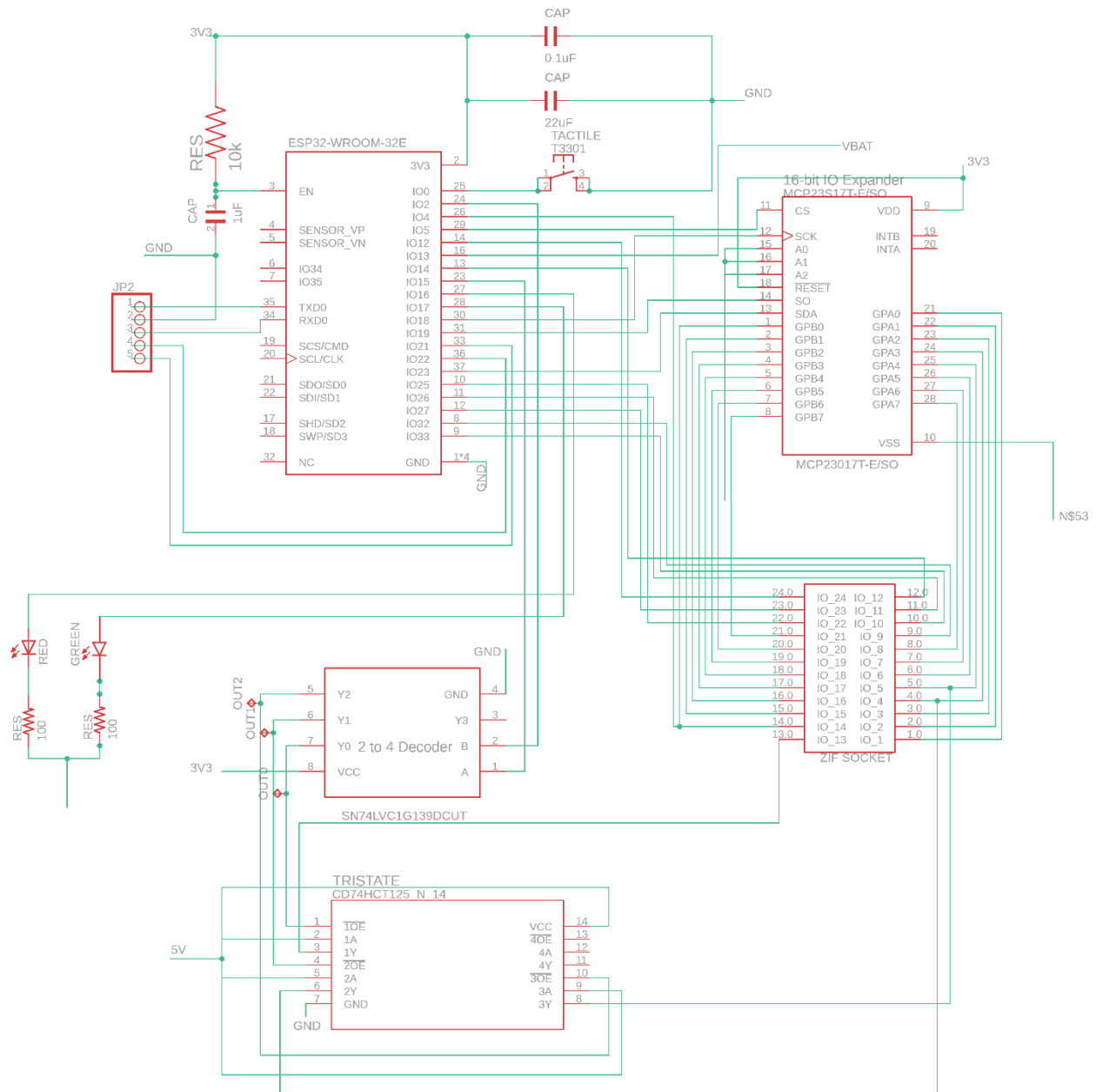


Figure 13: Eagle Board Layout of Automated IC Chip Tester

## Control Unit Eagle Schematic



### Figure 14: Eagle Schematic of Control Unit