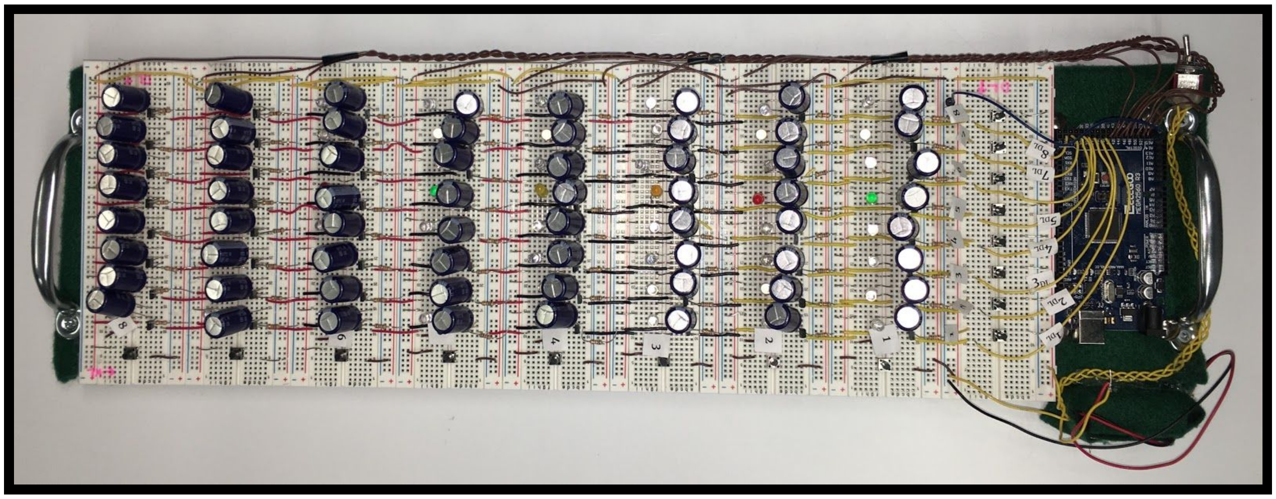
Large Scale DRAM Array Model

# User Manual

Revised May 2018



Sponsored by Micron Technology, Inc.

## Contact Information

### *DRAM Engineers*

| | | |
|---|---|---|
| Jinming Yang | e-mail: jy345@nau.edu | ph: +86 139.9088.6835 |
| Zeyu Zhang | e-mail: zz74@nau.edu | ph: +86 156.5666.0507 |
| Demetria Shepherd | e-mail: dcs245@nau.edu | ph: +1 602.574.7045 |
| Colby Weber | e-mail: ccw95@nau.edu | ph: +1 623.628.8708 |
| Abdulrahman Alqahtani | e-mail: asa266@nau.edu | ph: +1 202.568.5218 |

### *Client*

| | |
|---|---|
| Daniel Eichenberger | e-mail: deichenberge@micron.com |

### *Capstone Mentor*

| | |
|---|---|
| Ashwija Korenda | e-mail: Ashwijakorenda@nau.edu |

### *Faculty Mentor*

| | |
|---|---|
| Julie Heynssens | e-mail: julie.heynssens@nau.edu |

### *DRAM Engineers website*

https://www.cefns.nau.edu/capstone/projects/EE/2018/DRAMBoard/index.html

### *Micron Technology website*

https://www.micron.com/

## Table of Contents

## 1.0 Introduction

The following is a detailed user guide for the 8X8 DRAM Array model designed by the DRAM Engineers of the NAU Undergraduate Capstone team. The purpose of this model is for Micron Technology to use this as a simulation of the functions of DRAM, such as writing to, reading from, and refreshing memory cells within the DRAM Array. This model is intended to be used by Micron recruiters so they may show the functions of DRAM to prospective engineering employees at NAU's career fairs. It provides the recruiters an alternative to just showing DRAM chips on printed circuit boards.

### 1.1 Scope and Purpose

The need for a Large Scale model of a DRAM Array was made clear by Micron Technology's desire for a model to demonstrate to prospective employees of Micron's at NAU career fairs. Currently, Micron Recruiters have no way of demonstrating the operations and components that comprise their product that brings in a majority of the company's revenue.

So, the purpose of this model is

- To show the functions of DRAM
- To show the structure of a DRAM memory cell
- To show how a grid connected array is controlled

### 1.2 Model Functionality

The model accomplishes all three purposes outlined. The user is able to perform certain operations on the array, such as a computer would in a real DRAM chip. Using grid-connected push buttons, when the user presses one of the vertical buttons (wordlines) and one of the horizontal buttons (digitlines), then a cell has been selected to perform a "Write" action upon. The user can control what is written to the array by using a toggle switch to select what to write, a logic "1" or a "0", as shown in Caption 1 of Figure 1. When both desired buttons are pressed and a “1” is selected to be written, the LED/light indicator on that cell will brighten, and if a “0” is selected, the light on the cell will immediately dim, shown in Captions 2 and 3 of Figure 1. It can also be observed from Figure 1 that a cell is being manually charged in Caption 5.

The model is also a visual representation of DRAM, as much as it is a simulated representation. The user can visually see the construction of the array and how each memory cell is made up of an m-bit cell, a common type of memory cell structure used in this model, shown in Caption 4 of Figure 1. The components making up each memory cell include a storage capacitor, NMOS transistor, LED, and resistor, which will all be visible to an Electrical Engineer from looking directly down upon the array, shown in Captions 4a-4c of Figure 1.

Finally, the array is arranged in a grid-connected format. This means that the all 64 cells can be accessed independently using a combination of 16 push buttons. 8 buttons represent digitlines and another 8 represent wordlines which together control the two inputs of each cell. When a horizontally connected digitline button is pressed, and a vertically connected wordline button is pressed, that will correspond to an individual cell to which the user can write to, as featured in Figure 1. This grid-connected configuration is how an actual DRAM Array is connected on a chip.

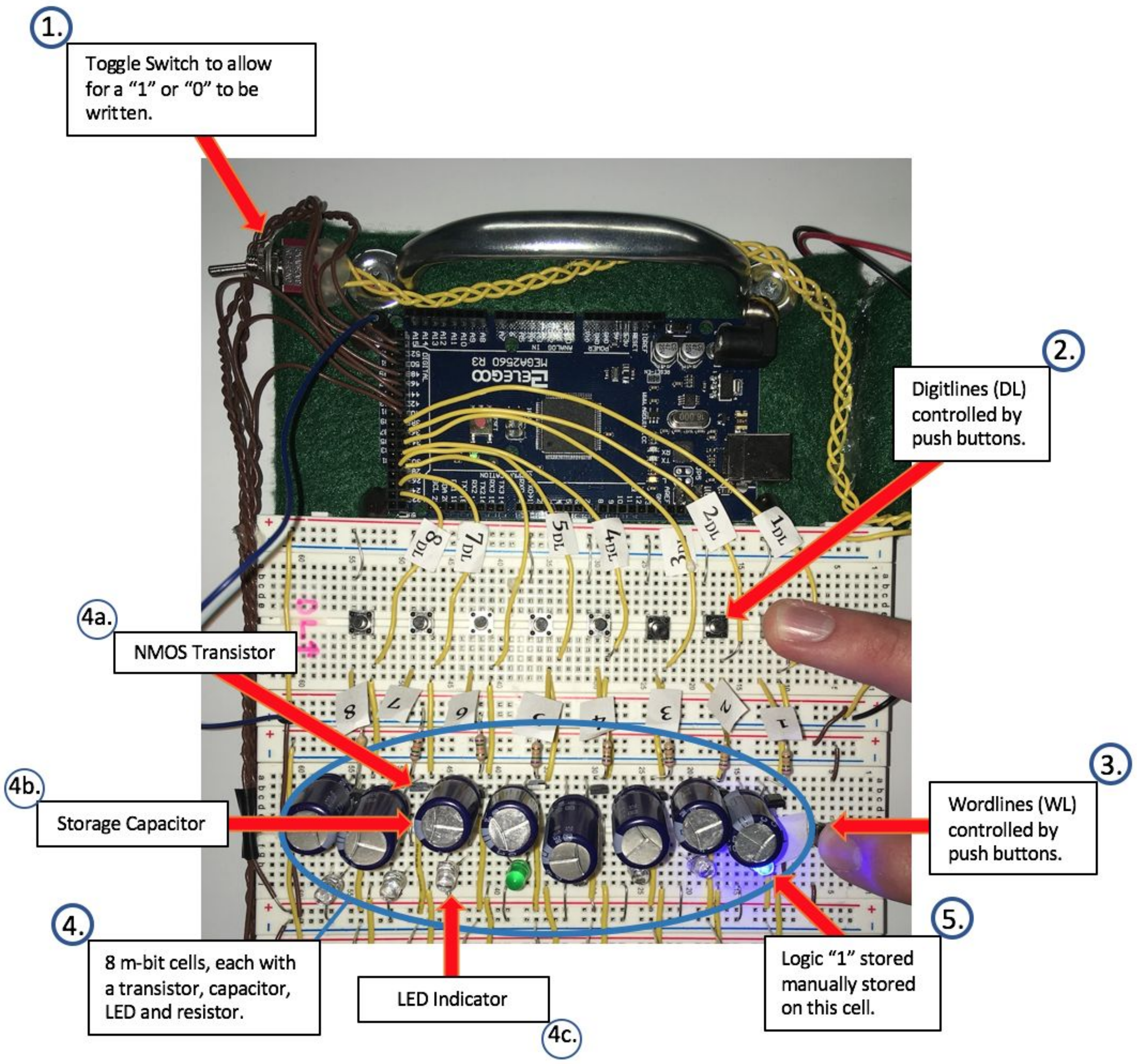

1.
Toggle Switch to allow for a "1" or "0" to be written.
2.
Digitlines (DL) controlled by push buttons.
4a.
NMOS Transistor
3.
Wordlines (WL) controlled by push buttons.
4b.
Storage Capacitor
4.
8 m-bit cells, each with a transistor, capacitor, LED and resistor.
LED Indicator
4c.
5.
Logic "1" stored manually stored on this cell.

*Figure 1: Detailed top down view of the DRAM Array model.*

## 1.3 Acknowledgements

DRAM Engineers would like to thank Micron Technology for providing the Northern Arizona University College of Engineering, Forestry, and Natural Sciences with this capstone project for the 2017-2018 academic year. We would also like to thank our Capstone Mentor, Ashwija Korenda, and our Faculty Mentors, Julie Heynssens, Dr. David Scott, and Dr. Kyle Winfree for their support and guidance with this project.

## 2.0 Background Information

In order to fully understand this project, it is necessary to familiarize oneself with the "heart" of the DRAM array, the m-bit cell. This is a commonly used type of memory cell used within the semiconductor memory field. It can be found storing data in a multitude of devices such as modern computers, smartphones, and many other Internet of Things devices. This section is provided to inform how an m-bit cell operates, as well as how the team went from the theoretical m-bit schematic, to the actual used in the model and why certain modifications were made.

### 2.1 Theoretical m-bit

The m-bit schematic is the main working component of this model. The team first examined a theoretical depiction of an m-bit schematic as seen in Figure 2. The circuit may seem complex at first glance, but its mode of operation is fairly simple. The NMOS transistor, denoted M1, acts as a switch with two inputs shown as a digit-line and word-line. A storage capacitor, C1, is connected to the output side of M1. When the wordline and digitline of M1 are in a high state, meaning a voltage pulse is being applied on both lines, then M1 creates a path for data to flow to C1. The information written will be in the form of a logic "1" or "0". C1 is always theoretically pre-charged to a $V_{CC}/2$ state, which is just half of the overall voltage being supplied to the array. This allows for a quick transition to either pulling the capacitor to the full $V_{CC}$ writing a "1", or to ground writing a "0". This was the basis for the project and the team's starting point in constructing a 64 m-bit array. From here, this schematic was built, tested and modified to fit the requirements of our client, Micron Technology.

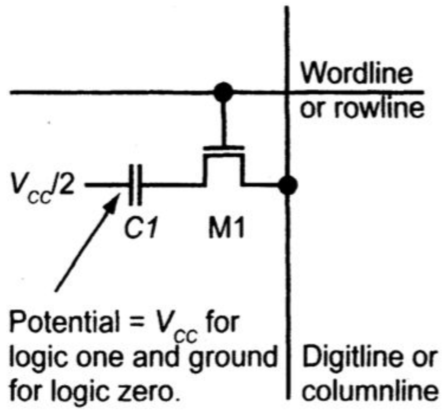

Wordline or rowline
$V_{CC}/2$
C1
M1
Potential = $V_{CC}$ for logic one and ground for logic zero.
Digitline or columnline

Figure 2: Theoretical m-bit cell schematic [1].

## 2.2 Actualization and Testing

After initial testing, the group finalized the schematic shown in Figure 3. The circuit shown still retains the original structure of an m-bit cell of 1 transistor and 1 capacitor for data storage. Digit and wordlines are still connected at the two critical inputs of the transistor, the source and gate, but are now physical

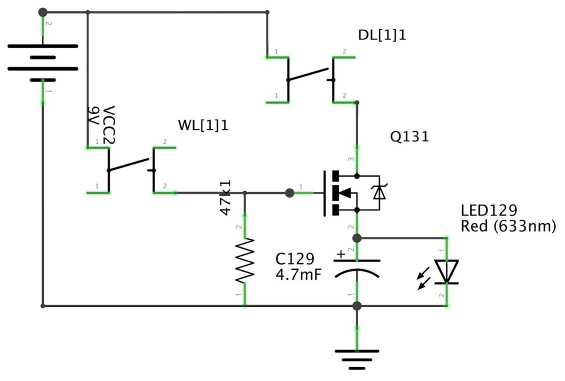


Figure 3: Actual schematic used in the model.

inputs in the form of 4-pin push buttons. Both are powered by an external 9V battery source and when both are pressed, the transistor allows a path for current to flow and in turn a voltage is stored on that 4.7mF capacitor. With such a large sized capacitor, it allows for a slow discharging rate which is the nature of DRAM to be volatile and lose charge over time. To demonstrate this, an LED was put in parallel with each capacitor of each cell which will brighten when the capacitor is fully charged, and then slowly dim as charge begins to fade from the capacitor.

A resistor was also added to the circuit which extends from the gate of the transistor connected to the wordline ( WL[1] ), to ground. The resistor was implemented to prevent any high floating gate voltages the team observed in testing. These floating gate voltages were a result from firing initial firing upon the wordline, which would send a voltage and open up the gate of the transistor allowing current to flow to the capacitor, but due to the capacitor remaining charged, these gates would be left open until the capacitor is fully discharged. So, this means that essentially any wordline would remain high for several minutes after initial firing allowing for multiple cells to be accessed when firing upon a single digitline row. The resistor to ground now only allows for the wordline to remain high as long as the button is pressed, while also still being able to store charge on the capacitor.

### 2.3 Grid-Connected Configuration

Once the schematic from Figure 2 was finalized in Figure 3, this circuit was used in the construction of the final, 64 m-bit array. Appendix A shows the full construction of the DRAM Array in a grid-connected 8X8 configuration; 8 wordlines and 8 digitlines are able to independently control each one of the 64 cells. Each wordline button is connected to 8 m-bit cells aligned in a horizontal row at the transistor gates. Similarly, each digitline button is connected to 8 m-bit cells arranged vertically in a column at the source sides of each transistor. This was repeated 7 more times both control lines, creating a total of 8 wordlines and 8 digitlines, as shown in the finalized DRAM array schematic in Appendix A.

## 3.0 Start-up of the Model

The DRAM array model consists of two main subsystems which include the actual 8X8 array and microcontroller, both mounted on a carrying board for ease of transport.

To operate the model, the user will need:

- The entire model with the DRAM array, microcontroller, battery, and toggle switch.
- The provided USB cable connecting the microcontroller to a laptop/computer.
- The software provided by DRAM Engineers.
- A working laptop/computer with USB ports.
- Arduino Genuino software which can be downloaded at: **https://www.arduino.cc/en/Main/Software**

## 3.1 Preparing to use the Model Manually

The DRAM array is powered by an external power source in the form of a 9V battery. This battery is contained within a sleeve attached to the carrying board and requires no help from the user to power the board aside from

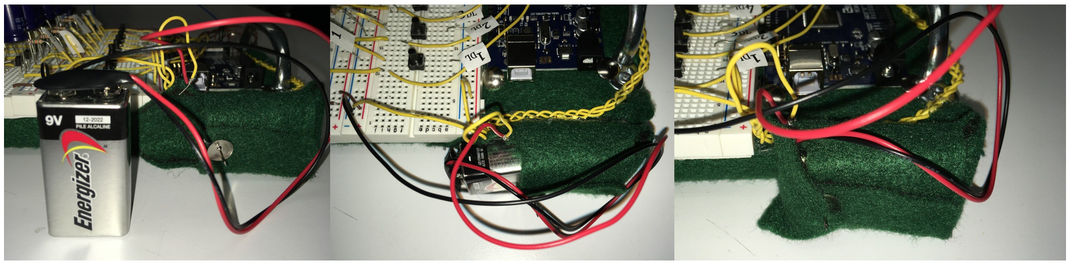


Figure 4: Attached battery compartment.

regular battery replacement (see 5.0 Maintenance). Using the attached pin on the sleeve, the user can easily remove and replace the battery as shown in Figure 4. The battery is connected to a toggle switch, also attached to the carrying board, which can toggle between $V_{CC}$ (9V) and ground so the user may
choose to write either a "1" by toggling to $V_{CC}$, or a "0" by toggling to ground, as shown in Figure 5.

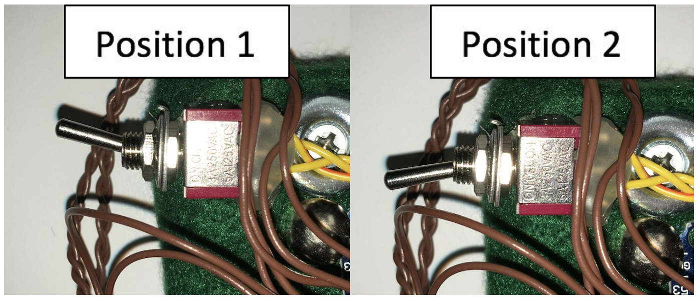


Figure 5: Attached toggle switch; first position writes "0", second position writes "1".

## 3.2 Preparing to use the Model via Software

The model can also be controlled using software installed on the provided Arduino Mega microcontroller attached to the board. To use software, the user
will need to first connect the microcontroller to their laptop or computer of choice, with the Arduino Genuino software installed listed in section 3.0, using the provided USB cable. Then, the user will need to run the program provided by DRAM Engineers and open the Serial Monitor located in the top left corner of the Arduino Genuino software, as shown in Figure 7. Once the Serial Monitor is opened, the user will be prompted to enter a digitline and wordline address, as well as be provided a method to refresh the array via software.

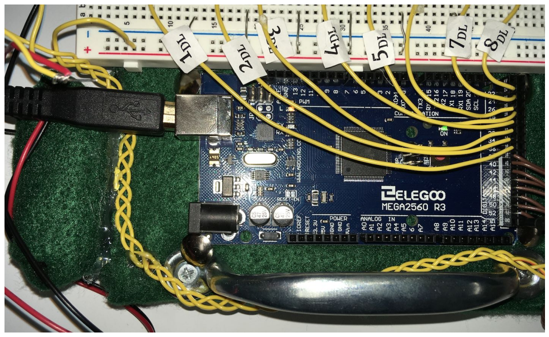
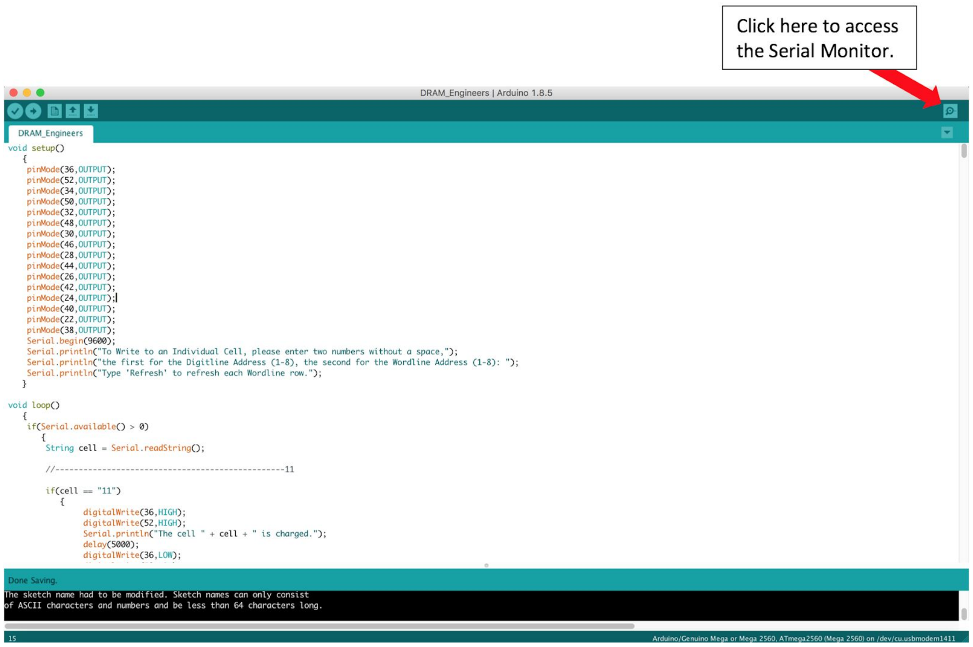


*Figure 6: Arduino Mega Microcontroller attached on the board.*


Click here to access the Serial Monitor.
DRAM_Engineers | Arduino 1.8.5
DRAM_Engineers

```
void setup()
  {
   pinMode(36,OUTPUT);
   pinMode(52,OUTPUT);
   pinMode(34,OUTPUT);
   pinMode(50,OUTPUT);
   pinMode(32,OUTPUT);
   pinMode(48,OUTPUT);
   pinMode(30,OUTPUT);
   pinMode(46,OUTPUT);
   pinMode(28,OUTPUT);
   pinMode(44,OUTPUT);
   pinMode(26,OUTPUT);
   pinMode(42,OUTPUT);
   pinMode(24,OUTPUT);
   pinMode(40,OUTPUT);
   pinMode(22,OUTPUT);
   pinMode(38,OUTPUT);
   Serial.begin(9600);
   Serial.println("To Write to an Individual Cell, please enter two numbers without a space,");
   Serial.println("the first for the Digitline Address (1-8), the second for the Wordline Address (1-8): ");
   Serial.println("Type 'Refresh' to refresh each Wordline row.");
  }

void loop()
  {
   if(Serial.available() > 0)
     {
      String cell = Serial.readString();

      //-------------------------------------------------11

      if(cell == "11")
        {
             digitalWrite(36,HIGH);
             digitalWrite(52,HIGH);
             Serial.println("The cell " + cell + " is charged.");
             delay(5000);
             digitalWrite(36,LOW);
```

```
Done Saving.
The sketch name had to be modified. Sketch names can only consist
of ASCII characters and numbers and be less than 64 characters long.
```

```
15                                   Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on /dev/cu.usbmodem1411
```

*Figure 7: Accessing the Serial Monitor in the software provided.*

## 3.3 Transporting the Model

The entire model is mounted on a carrying board with handles on both sides for a portability ease, shown in Figure 8. The entire model measure 20" x 7" x 6" (L x W x H) and weighs approximately 5 lbs.

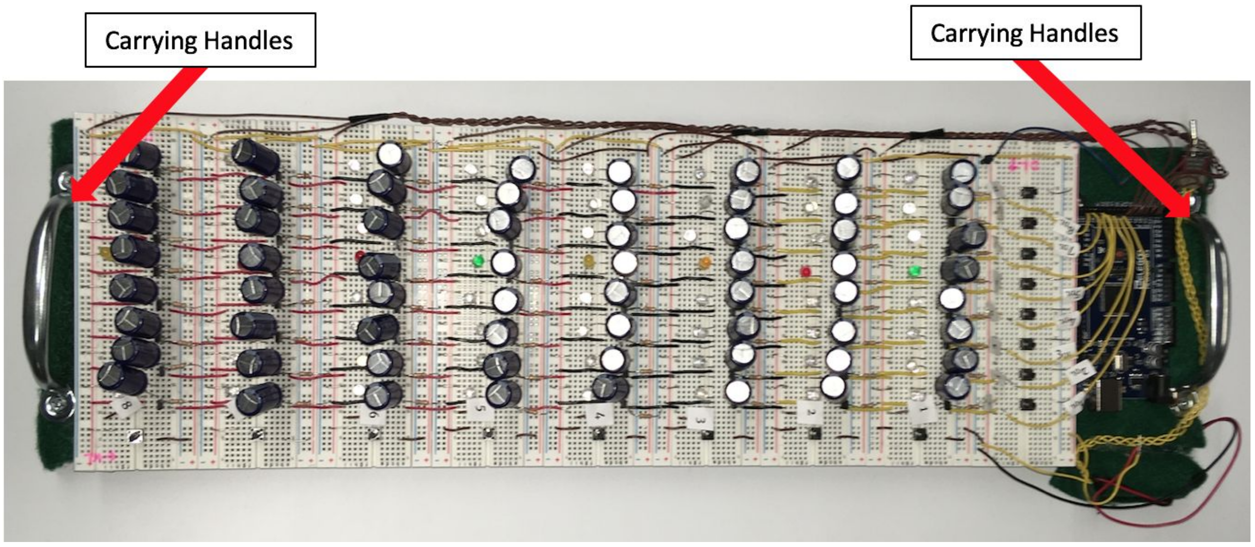


*Figure 8: Carrying handles allow for ease of mobility.*

Please be advised:

- If the model is transported outside, it should be covered from any rain or snow as that will critically damage the DRAM array and microcontroller.
- Do not drop the model if at all possible.
- Always pick up the model by the provided handles in the upright position with the blue capacitors facing upward.

## 4.0 Configuration and Use

This DRAM model is to simulate the basic RAM functions which allows users to write and read information, with the advantage of rapid reaction, low cost, and small size. The functions of this model are divided to three parts which are information writing, information reading and refreshing respectively. This DRAM model can only achieve parts of the more advanced functions because this model only contains the DRAM array. The model does not contain other components of DRAM such as the Sense Amp or other peripheral circuitry. However, this completed basic DRAM array could be used to develop the deep functions moving forward.

## 4.1 Writing

The first basic function is to write the signal to certain cells in DRAM array by charging the capacitor, which would be shown by LED connected in parallel.

Instructions for Writing to the Array:

- Option #1: Write Manually
    1. Position the toggle switch into of the two positions listed in Section 3.1, selecting whether a "1" or "0" will be written.
    2. Using the provided numbering flags as a guide, the user will then press a single digitline and wordline button to write to a selected cell.

- Option #2: Write via Software
    1. Connect the Arduino Mega microcontroller to a laptop/computer via the provided USB cable.
    2. Download/open the Arduino Genuino software and open the provided program titled "DRAM_Engineers".
    3. Run the program and open the Serial Monitor to show the corresponding instructions.
    4. Enter a corresponding digit and wordline address as prompted by the Serial Monitor to write to a memory cell.

### 4.1.1 Digital Signal by Manual Control

Digital signals are widely used in the field of computer science. CPU (Central Processing Units), microprocessors and microcontrollers can only receive and handle these digital signals.

There are only two states of a digital signal; a high state is represented by a logic "1" and a low state corresponds to a logic "0". In this DRAM array model, a capacitor on a cell being charged means that this cell is in a high state. If the user provides voltages on both digitlines and wordlines continuously, this cell will keep the high state for as long as both buttons are pressed. Otherwise, the capacitor will discharge, which means the state of this cell is forced to low.

There is an inner 9V battery source provided to fire on the digitlines and wordlines manually. The push buttons on the digitlines and wordlines give the user more choices on how to select a cell they would like to perform a write action upon. Via manual control, only digital signals can be stored in to the memory cells. It is allowed to write to multiple cells at the same time but, might not be as brightly shown by the LEDs because the divided voltage may not fully satisfy their voltage requirement to fully brighten.

When the voltage source disconnects from the digitlines and wordlines, the digital signal stored in the cell will fade away in a short period, so manual-controlled DRAM array is not allowed to transmit data over long periods of time.

#### 4.1.2 Message Stored by Microcontroller

There are 64 cells in the DRAM array, which means users could store 64 bits in the array by using the microcontroller. Based on the decoding/command tables, users could translate a message using simple bit allocation done by the Arduino Mega, then store the bits into the DRAM array. This part will need a seperate completed software code, so it will be developed in the future work.

## 4.2 Reading

Reading data is one of the important functions of a DRAM array. DRAM is a kind of semiconductor memory, so when users store digital signals in the array, ideally others could read the correct information given certain decoding/command tables as mentioned below. For this model, no matter what kind of information is stored in the DRAM array, users have to use the microcontroller or LEDs to read the information from array.

Instructions for Reading from the Array:

- Option #1: Read Manually
    1. Whether the information was stored via software or manual operation, the user can visually read out if a “1” is stored on a cell by observing the LED on that cell brighten. A “0” stored will immediately dim the LED.

- Option #2: Read via Software
    1. This can only be done if the information written was done via software.
    2. When the software has written to the selected cell, a message on the Serial Monitor will read “The cell is charged”.
    3. The microcontroller will then discharge the selected cell after 5 seconds and will read “The cell is discharged”.

### 4.2.1 Decoding Information

Decoding is a commonly used example of how digital signals are used by the DRAM array. There should be a “rule” which is known by both the receiver and sender of the message stated in Section 4.1.2. Based on the special translating rule, the senders could charge a certain cells’ address, to store the digital signal on the array. When the receiver observes this digital signal on the array, they would use the microcontroller to read the certain cells which have already charged, then translate this cryptographic information based on the translation rule.

### 4.2.2 Command Code Table

The theory of command code table is almost the same as with the decoding translation rule. However, there is a distinct difference between the decoding and the command code table; the matching command is public but the rule of decoding is private.

Actually, with a built-in program ( java, C++, python, etc. ) the charged DRAM array is a kind of controller ( to be developed ). Different commands will match the different addresses charged. Users could achieve  control of the PC through charging the DRAM array according to a custom made command code table. The theory is that when the users charge the DRAM array, the microcontroller could read the digital signals and send those signals to the processor. Then, the processor will execute the specific command based on the translation rule encoded into a program, consequently, achieving the purpose of control of the PC automatically.

## 4.3 Refreshing

The industry standard for refreshing a DRAM array is 64ms [1]. In order to simulate this function on the model, software was developed to allow the user to refresh the model, column by column each controlled by a digitline.

Instruction for Refreshing the Array:

1. Connect the Arduino Mega microcontroller to a laptop/computer via the provided USB cable.
2. Download/open the Arduino Genuino software and open the provided program titled “DRAM_Engineers”.

3. Run the program and open the Serial Monitor to show the corresponding instructions.
4. Enter “Refresh” into the Serial Command Line and observe the array refresh column by column.

## 5.0 Maintenance

Some forms of semiconductor memory have a volatile nature, which requires a constant power source to hold stored information data on the memory array. This is the case for DRAM, and due to its volatility, the model and all electrical components will experience constant charging and discharging cycles. Over time, this will lead to the failure of some of these key electrical components. This section provides general tips and information to continuously use and maintain the DRAM model broken down into two different aspects, the software and hardware.

### 5.1 Software

As mentioned before, the DRAM array can be controlled in two different operations: manually and through software. Using software, the user is able to input a memory address and have the microcontroller send digital pulses to store charge on the selected address. DRAM Engineers will provide with this report the necessary program to use software to control the array. The code is relatively maintenance free. The only maintenance in required maybe regular updating of the Arduino Genuino software used to create and run the software, listed in Section 3.0. Figure 6 shows an example of the code structure for the DRAM array model.

```
if(cell == "##")
// the 1st digit is DL address, and 2nd is the WL address.
// DL = digit-line; WL = word-line
           {
digitalWrite(pinMode,HIGH); // pin on the Arduino set high
digitalWrite(pinMode,HIGH); // pin on the Arduino set high

Serial.println("The cell " + cell + " is charged.");
// A message tells user the cell is being charged.

delay(5000); // Five second delay

digitalWrite(pinMode,LOW); // pin on the Arduino at low
digitalWrite(pinMode,LOW); // pin on the Arduino at low

Serial.println("The cell " + cell + " is discharged.");
// A message tells user the cell is being discharged.
               }
```
:

*Figure 9: An example of the DRAM model code structure.*

For information on coding in the Arduino software system, please refer back to company's website at **https://www.arduino.cc/en/Guide/HomePage.**

## 5.2 Hardware

There are so many components on this project that certain parts might need to be maintain so often. These include:

- <u>LEDs:</u> If the LEDs burn out, they will need to be replaced with similar LEDs requiring a voltage of 9V or less to power them. .
- <u>Capacitors:</u> The capacitors might stop holding charges after so many charge cycles. In that case replace capacitors as needed in order to observe the element's state of charging and discharging via the LED for

approximately 5 seconds. Use capacitance values of 4700uF or higher and are able to store up to 6.3V.

- __NMOS Transistors:__ If it is observed that multiple cells are being accessed at once, then a transistor has probably met its product end of life. Please replace any transistors with standard breadboard sized NMOS transistors. The team recommendation is the ZVN3306A model transistors.
- __Resistors:__ If multiple cells are still being accessed after transistor replacement, then it is likely that a resistor has met the end of its product life. Replace any resistor with a value of 47kΩ or higher.
- __Push Buttons:__ From observations, the push buttons can sometimes lift from the breadboard accidently, so be careful to replace the push button back as to not break off one of the 4 pins. If that does occur, please replace with one of the provided extra push buttons, or just another 4-pin push button.
- __Arduino Mega Microcontroller:__ The microcontroller is fairly robust with built in over-current protection as to protect against electrical shorts. If the microcontroller does no longer power on when connected via the USB cable, please replace with another Arduino Mega Microcontroller but before replacing, be sure the USB cable is not at fault by replacing that component first.
- __9V Battery:__ If all of the cells on the array barely brighten when using the push buttons, the battery is probably dead so please remove the battery from the sleeve, carefully disconnect the battery terminals and reconnect a new 9V battery.

## 6.0 Troubleshooting Operations

- An LED on a memory cell doesn't light as expected:
    1. This LED might be burnt out, please change with another LED to confirm the problem, then replace.
    2. If the LED is not the problem, the battery might be dead so replace the 9V battery.

    3. If the battery is not the problem, ensure that all wires are securely pressed down on the breadboards and try again. Please refer to Appendix A for wiring instruction.
    4. If the problem persists, ensure the Arduino Mega is grounded via the long blue wire extending from microcontroller to ground on the array.

- "Refresh" function doesn't work as expected in the software
    1. If there is something wrong with uploading, please check the right board type and COM port is selected, which should be Arduino Mega. This will be under "tools" in the Arduino Genuino software.
    2. Make sure the code is correct. Sometimes, the code may be changed by mis-touching on keyboard without consciousness. Close the Arduino Genuino software without saving the program.
    3. Type the right word into the Serial Monitor, which should be "Refresh".
- Multiple cells are accessed at once when pressing a single digit and wordline button
    1. Check to make sure the long blue wire from the Arduino Mega is connected to a ground bus on one of the array's breadboards.
    2. Ensure that the digitline and wordline connections from the Arduino Mega to the array are properly connected and secure.
    3. Check each resistor on each cell to ensure they are all connected from the gates of each transistor to ground. Refer to Appendix A for a detailed overview.
    4. Always be sure to power the array using one 9V battery. More than 10V will begin to damage the model.
- Trouble uploading "DRAM_Engineers" program to the Arduino Mega
    1. Under "Tools" in the Arduino Genuino software, be sure the correct board is selected, "Arduino/Genuino Mega or Mega 2560".
    2. Under "Tools", ensure the correct "Port" is selected. You know you have selected the correct port when the board name from step 1 appears next to the COM number.

## 7.0 Status of Planned Features: Work Breakdown Structure (WBS)

In this section of the manual, each team member will be discussing their expectations versus what was accomplished in the project.

### 7.1 Abdulrahman's WBS Explanation

Appendix B, is where my WBS located and states all of the tasks I was responsible and working on with my teammates during the academic year. Main Activity in My WBS Includes:

- Designing of the DRAM Array with Four subtasks:
    - Finalized circuit design
    - Working 8X2 Prototype
    - Finalized the Arduino and programing
    - Finalized the entire model

All of the above tasks in my Activity have been done and they belong in the section below:

- The first subtask was to design the circuit which implemented all of the required parts. The deliverable for this task was a fully realized schematic. The level of completion is 100% on that subtask.
- The second subtask consisted of constructing a prototype which included building 16 cells in an 8X2 configuration; 8 wordline and 2 digitline and the team ensured it was fully operational and was what our client asked for. The deliverable for this task was to build the entire circuit and ensure the functionality of all 16 memory cells in terms of writing to each one independently. Based on that schematic we built the entire model. The level of completion is 100%.
- The third subtask was to test the Arduino. The Arduino needed to be able to take in word and digitline addresses corresponding to a selected cell as chosen by the user, which will light the LED to show the state of charging on the Capacitors. The deliverable of this task

was to make sure the Arduino could fully control one cell of the array. The level of completion is 100%.

- The last subtask was to Build and Test which implemented the Array and Microcontroller together. The Deliverable of this task was to test the entire system and make sure everything working as expected. The level of completion is 100%.

## 7.2 Colby's WBS Explanation

A requirement of successfully completing the project was that each team member needed to identify at least one main activity to complete. Below are my two main activities with accompanying tasks, also shown in Appendix C. The item lines following each activity denote deliverables.

Main Activities in the WBS:

- Designing the 8X8 DRAM Array
  - Finalized m-bit schematic.
  - Finalized 8X8 DRAM Array schematic.
  - Functioning 8X2 prototype.
  - Functioning 8X8 Array.
- Implementation of the Microcontroller
  - Flowchart displaying desired functionality.
  - Working prototype of code to access an individual cell.
  - Complete software program which can write to individual cells and also refresh the entire array, digitline by digitline.

All of the tasks above were completed successfully without compromise. The model accurately simulates the functions of DRAM by allowing the user to manually select whether to write a "1" or "0" using the toggle switch, then selecting the desired cell to write that data to by using the external digit and wordline push buttons. The user can also access the array via software programmed on the Arduino Mega microcontroller which takes in a memory cell address and then writes a "1" to the selected cell. The software also can refresh the entire model by entering the command, “Refresh”, into the Serial Monitor.

## 7.3 Demetria’s WBS Explanation

For this capstone project, each person in the team was responsible to complete a WBS with assigned task activities, and my WBS is located in the appendices section, Appendix D.

My WBS includes two activities and followed by other tasks assignments:

- Design the 8X8 (64 mbit cells) DRAM Array
    - Prototype/circuit Design
    - Testing on breadboard
    - Design the entire 64 mbit cells on breadboards
- Design a Carrying Tray for the 8X8 DRAM Array
    - Measure the full size of the DRAM Array
    - Completed “tray” design for the DRAM Array

In the designing stage of the DRAM array, almost every team member was responsible to meet and build the array during team meetings. This required a schematic of the mbits location followed by push buttons and toggle switch. To view the schematic, please refer to the appendices on appendix A. Also, testing the hardware of DRAM array was quite difficult yet minor to fix. We did run into some wire and electrical parts errors, but we managed to adjust the DRAM array. Overall this main activity is entirely 100% completed, and our client was satisfied of the functionality.

Lastly, designing a carrying tray for the DRAM array did include everyone’s input to ensure the correct size and usage. At first, our intention was to build a fitted “carrying case” for the DRAM, but due to time and cost we’ve decided to make it into a tray instead. Plus, a carrying tray did meet our team’s specification to make it easier for our client to hold/travel around.

Overall this main activity is entirely 100%, and our client was satisfied of the final design.

## 7.4 Jinming's WBS Explanation

My WBS is located in the appendices at Appendix E. Personally, I was responsible for two main activities with my fellow teammates, which are listed below:

- Implement Microcontroller
    - Determine functionality
    - Program
    - Ensure functionality
- Design Extra Functionality
    - Design and program a GUI (Graphical User Interface)

For the first activity, implementing the microcontroller was my responsibility followed by subtasks. We put those actions into a flowchart which gave us a clear direction to program on Arduino Mega. According we discussing, we finished programs. There were some problems in syntax for us in our programming processing. We found some example code on Internet. And we figured all problems out, and tested functions of our microcontroller connected to DRAM array after finishing coding. After all of this, I did a 100% completion level on this task activity.

For the second part, I tried to find a Arduino library which support the collection between the microcontroller and PC. I found a library at the first half of this academic year. But some functions in this library didn't work so well. I was not sure this library can satisfy all requirements of GUI. Because of rebuilding of DRAM array, I estimated time is not enough with potential problems in developing GUI. I stopped developing GUI. But we still have a way for user interacting with microcontroller on PC. Users can use the console in Arduino IDE to make the microcontroller perform functions, including accessing cells and refreshing.

## 7.5 Zeyu's WBS Explanation

I was responsible for the circuit design and software code. There were several different choices when we started working on the circuit design, because we could not know which one should be the final decision before we testing. In that case, we spent a lot of time researching and tested the circuits found in our research to determine a final design.

In Fall 2017, we were initially going to use the operation amplifier to increase the input voltage from the Arduino Mega to power all of the electronic components. However, this was a not well thought out idea because the completed logic of the amplifier and its voltage rails needing to be higher than the input voltage, this made the circuit not function as desired. Also, because the amplifier needed such a  large positive and negative voltage source to amplify the already low 5V from the Arduino Mega, it made no sense to use the amplifier if an external power source was required use it. So at the end of this semester, we were no longer considering the operation amplifier as a vital component in our DRAM array.

In Spring 2018, we had tried to build a prototype before we worked on the completed DRAM array. When we were doing the prototype, we experienced many problems because we misunderstood the basic working theory of the DRAM cell. We used the Arduino to provide the voltage for the DRAM cell, but there were a over current protection within the Arduino, which means if the voltage output is directly connected to ground without a suitable resistance, the circuit would NOT be shorted because the voltage would shut off due to the over current protection. So between the gate of transistor and ground, we added a resistor to prevent large floating gate voltages. Finally, the cell worked perfect and we used the same theory to do the 64 cell array and connected them with digitlines and wordlines together.

For the software code part, we just developed the basic write function of DRAM array. We could use the Arduino Genuino Serial Monitor to charge certain cells automatically by choosing the specific address. Also we could use Arduino to do the refresh, which would charge the DRAM each digitline by digitline. Currently, both of my main activities are 100% complete with no discrepancies.

## 8.0 Conclusion

The purpose of this capstone project was to create an interactive, simulated DRAM array and to show the operation of how volatile semiconductor memory works. Our client, Daniel Eichenberger (or Dan), requested this project to emulate DRAM functionality, such as writing to, reading from, and refreshing the memory array. Dan is a recruiter for Micron Technology and he wanted our DRAM model to become a teaching device to anyone interested in computer memory.

Our team has completed and accomplished most of the requirements asked by Dan, as well as met our own team specifications. We hope Dan will use our DRAM array model at future career fairs at NAU, showing Micron Technology's most profitable product in a way other than just having DRAM chips on display. This model will be able to show the three basic functions of DRAM to engineering students interested in pursuing a career with Micron.

During the whole academic year in capstone, we've learned so much from one others creativity, experiences, and knowledge. The team would like to thank Dan for making time to meet, mentor, and accept our DRAM array model as well as Micron Technology for providing NAU with a capstone project. We also would like to thank Dr. David Scott, Dr. Kyle Winfree, Julie Heynssens, and Ashwija Korenda for all of their guidance and support throughout this academic year. We wish the best of luck to our team members in their professional careers, and Micron engineers to continue enhancing memory storage for the future.

Thank you,
DRAM Engineers

[illegible] [illegible]

[illegible] [illegible] [illegible]

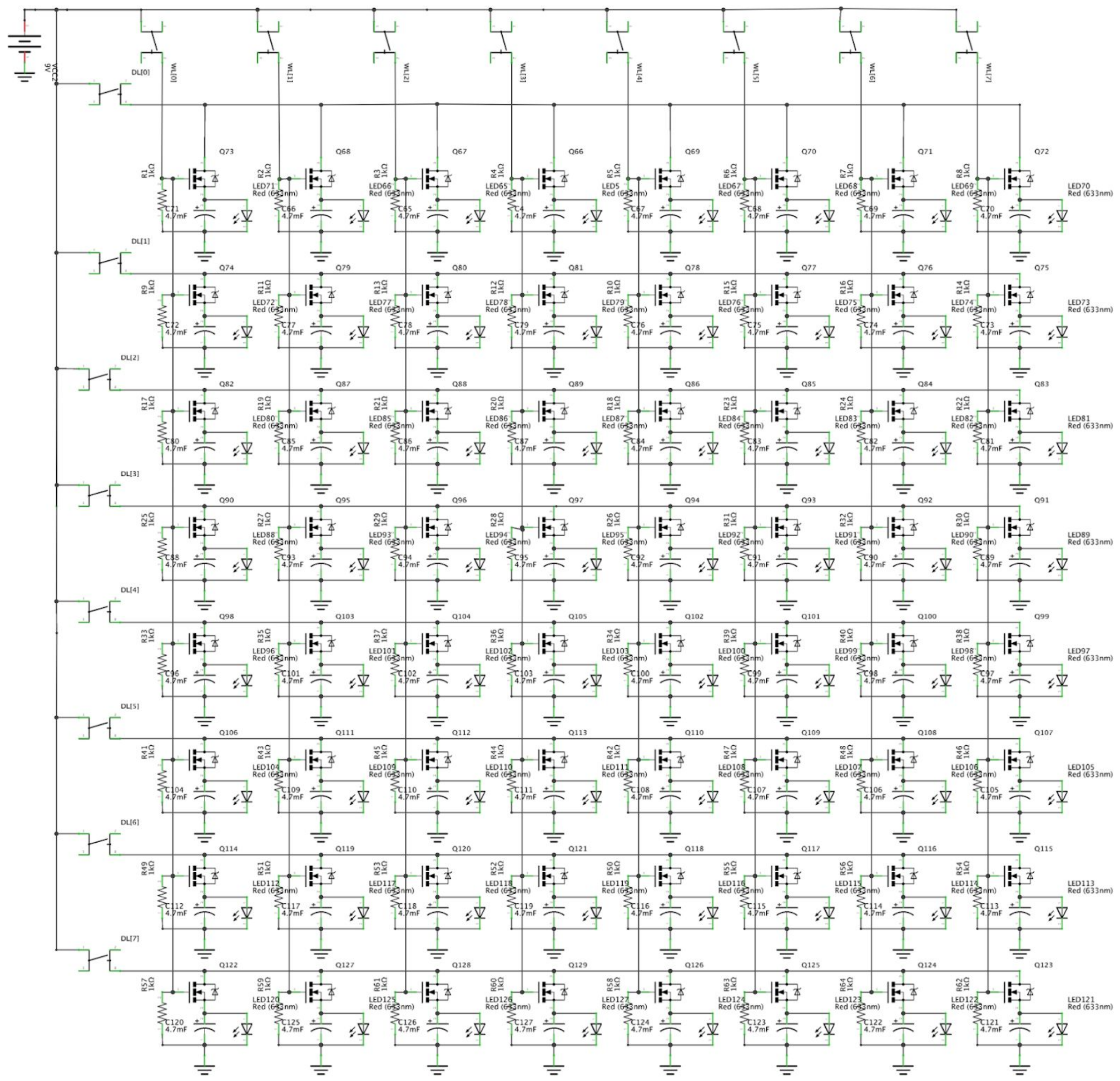
## 9.0 References

[1] B. Keeth, J. Baker, B. Johnson, and F. Lin, DRAM Circuit Design: Fundamental and High-Speed Topics. Hoboken: John Wiley, 2008.

## 10.0 Appendices

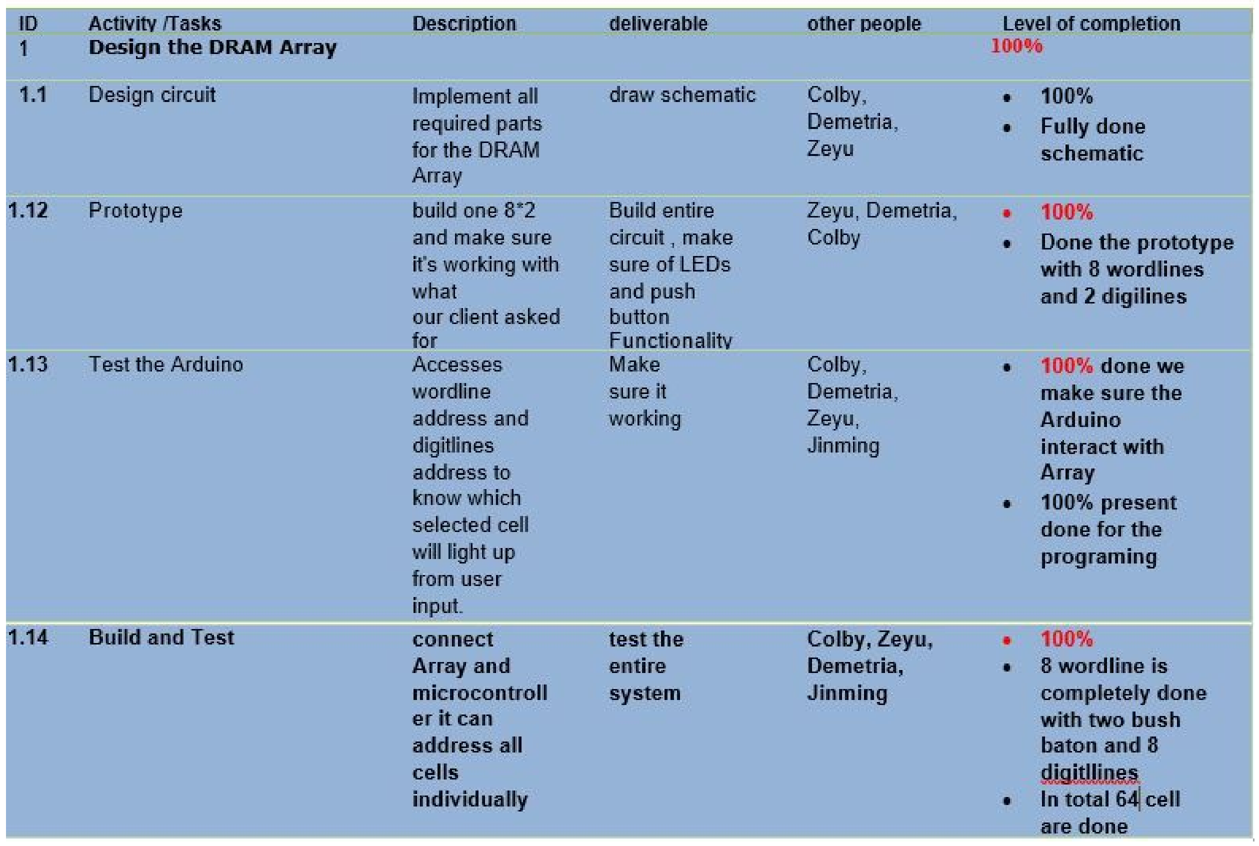
### Appendix A

Schematic of the 8X8 DRAM Array



## Appendix B

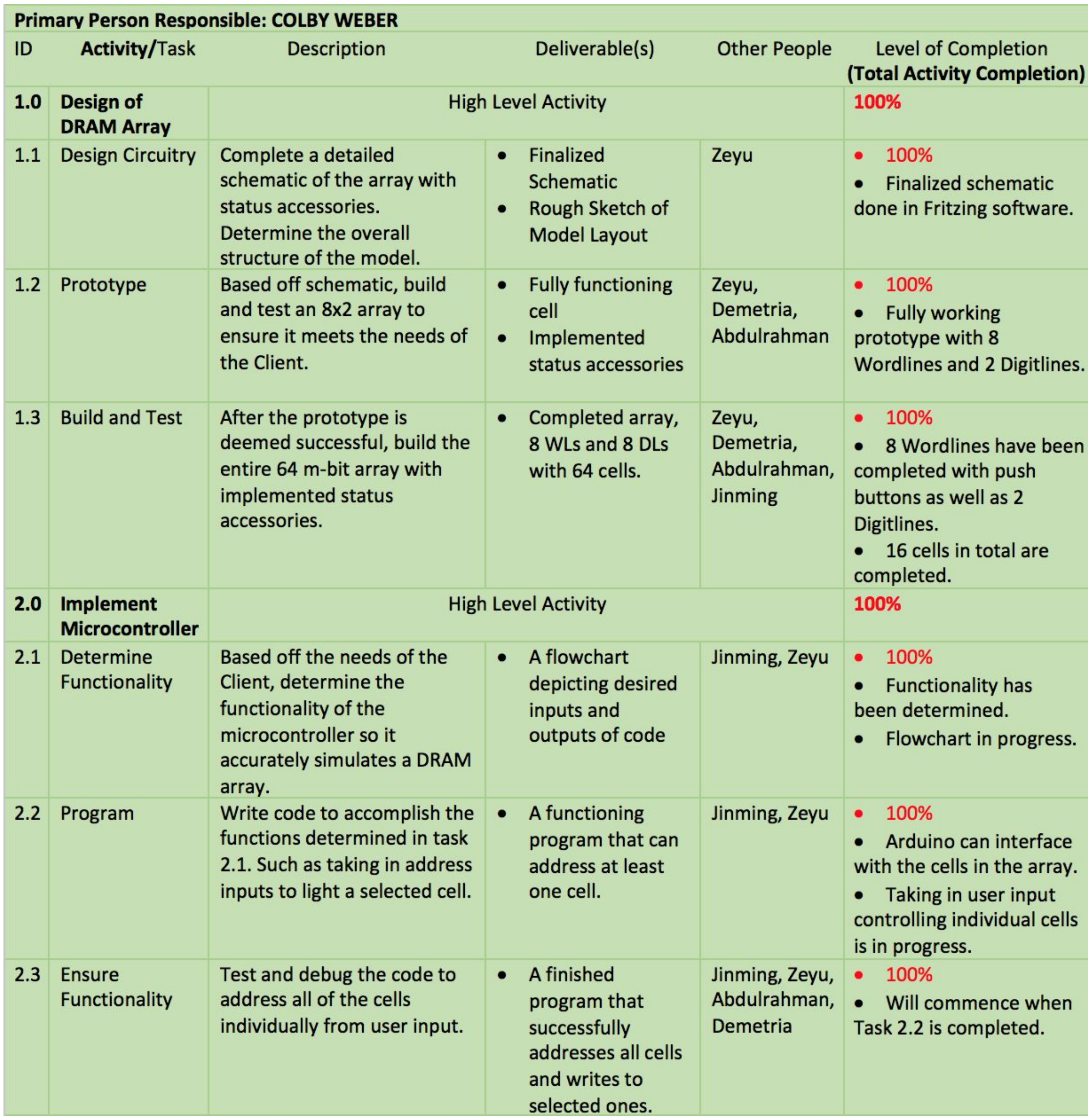
### Abdulrahman's WBS



| ID | Activity /Tasks | Description | deliverable | other people | Level of completion |
|---|---|---|---|---|---|
| 1 | **Design the DRAM Array** | | | | 100% |
| 1.1 | Design circuit | Implement all required parts for the DRAM Array | draw schematic | Colby, Demetria, Zeyu | • **100%** <br> • **Fully done schematic** |
| 1.12 | Prototype | build one 8*2 and make sure it's working with what our client asked for | Build entire circuit , make sure of LEDs and push button Functionality | Zeyu, Demetria, Colby | • **100%** <br> • **Done the prototype with 8 wordlines and 2 digilines** |
| 1.13 | Test the Arduino | Accesses wordline address and digitlines address to know which selected cell will light up from user input. | Make sure it working | Colby, Demetria, Zeyu, Jinming | • **100% done we make sure the Arduino interact with Array** <br> • **100% present done for the programing** |
| 1.14 | **Build and Test** | **connect Array and microcontroller it can address all cells individually** | **test the entire system** | **Colby, Zeyu, Demetria, Jinming** | • **100%** <br> • **8 wordline is completely done with two bush baton and 8 digitllines** <br> • **In total 64 cell are done** |

## Appendix C

Colby's WBS

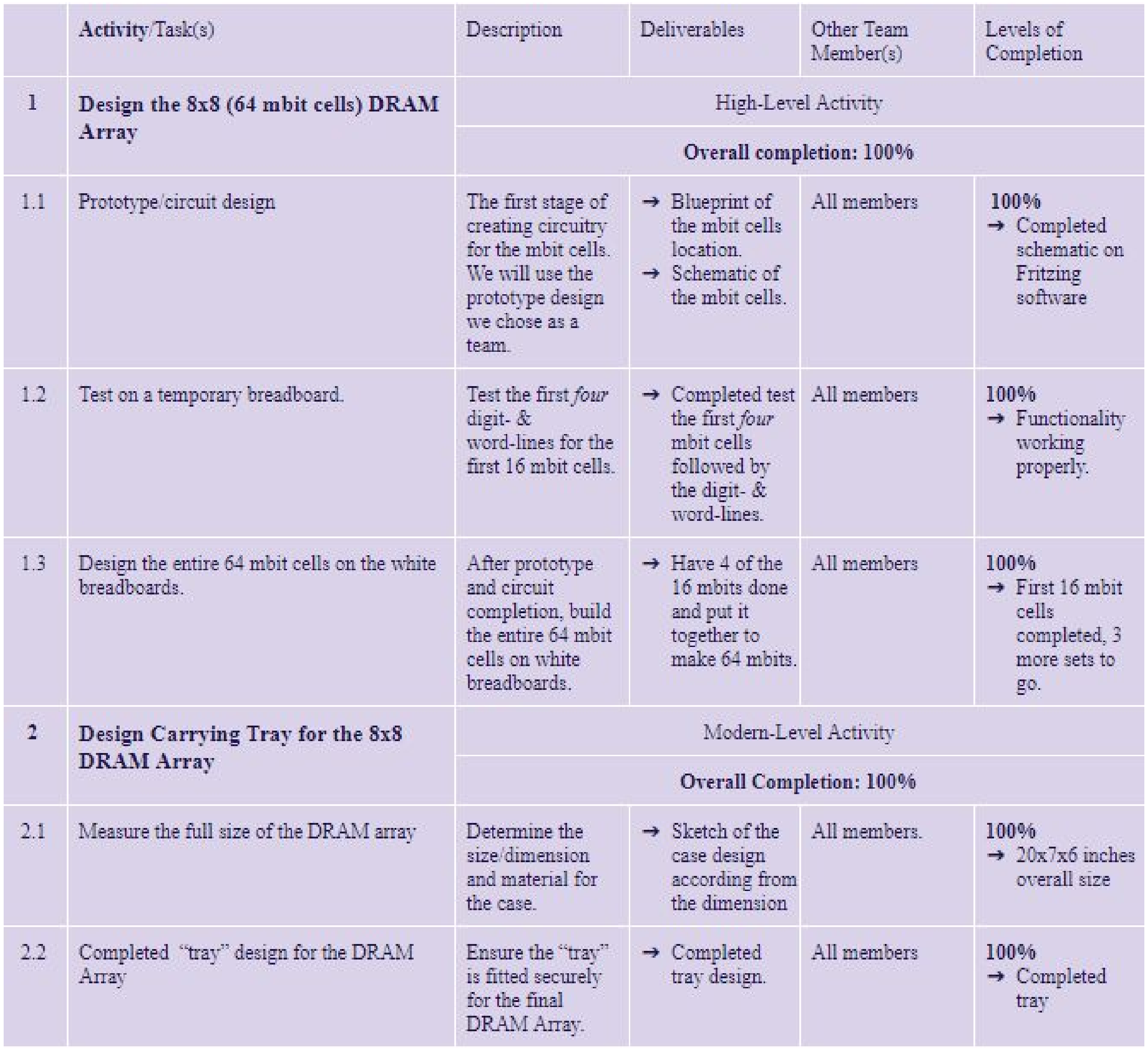


**Primary Person Responsible: COLBY WEBER**

| ID | **Activity**/Task | Description | Deliverable(s) | Other People | Level of Completion **(Total Activity Completion)** |
|---|---|---|---|---|---|
| **1.0** | **Design of DRAM Array** | High Level Activity | | | **100%** |
| 1.1 | Design Circuitry | Complete a detailed schematic of the array with status accessories. Determine the overall structure of the model. | • Finalized Schematic • Rough Sketch of Model Layout | Zeyu | • 100% • Finalized schematic done in Fritzing software. |
| 1.2 | Prototype | Based off schematic, build and test an 8x2 array to ensure it meets the needs of the Client. | • Fully functioning cell • Implemented status accessories | Zeyu, Demetria, Abdulrahman | • 100% • Fully working prototype with 8 Wordlines and 2 Digitlines. |
| 1.3 | Build and Test | After the prototype is deemed successful, build the entire 64 m-bit array with implemented status accessories. | • Completed array, 8 WLs and 8 DLs with 64 cells. | Zeyu, Demetria, Abdulrahman, Jinming | • 100% • 8 Wordlines have been completed with push buttons as well as 2 Digitlines. • 16 cells in total are completed. |
| **2.0** | **Implement Microcontroller** | High Level Activity | | | **100%** |
| 2.1 | Determine Functionality | Based off the needs of the Client, determine the functionality of the microcontroller so it accurately simulates a DRAM array. | • A flowchart depicting desired inputs and outputs of code | Jinming, Zeyu | • 100% • Functionality has been determined. • Flowchart in progress. |
| 2.2 | Program | Write code to accomplish the functions determined in task 2.1. Such as taking in address inputs to light a selected cell. | • A functioning program that can address at least one cell. | Jinming, Zeyu | • 100% • Arduino can interface with the cells in the array. • Taking in user input controlling individual cells is in progress. |
| 2.3 | Ensure Functionality | Test and debug the code to address all of the cells individually from user input. | • A finished program that successfully addresses all cells and writes to selected ones. | Jinming, Zeyu, Abdulrahman, Demetria | • 100% • Will commence when Task 2.2 is completed. |

## Appendix D

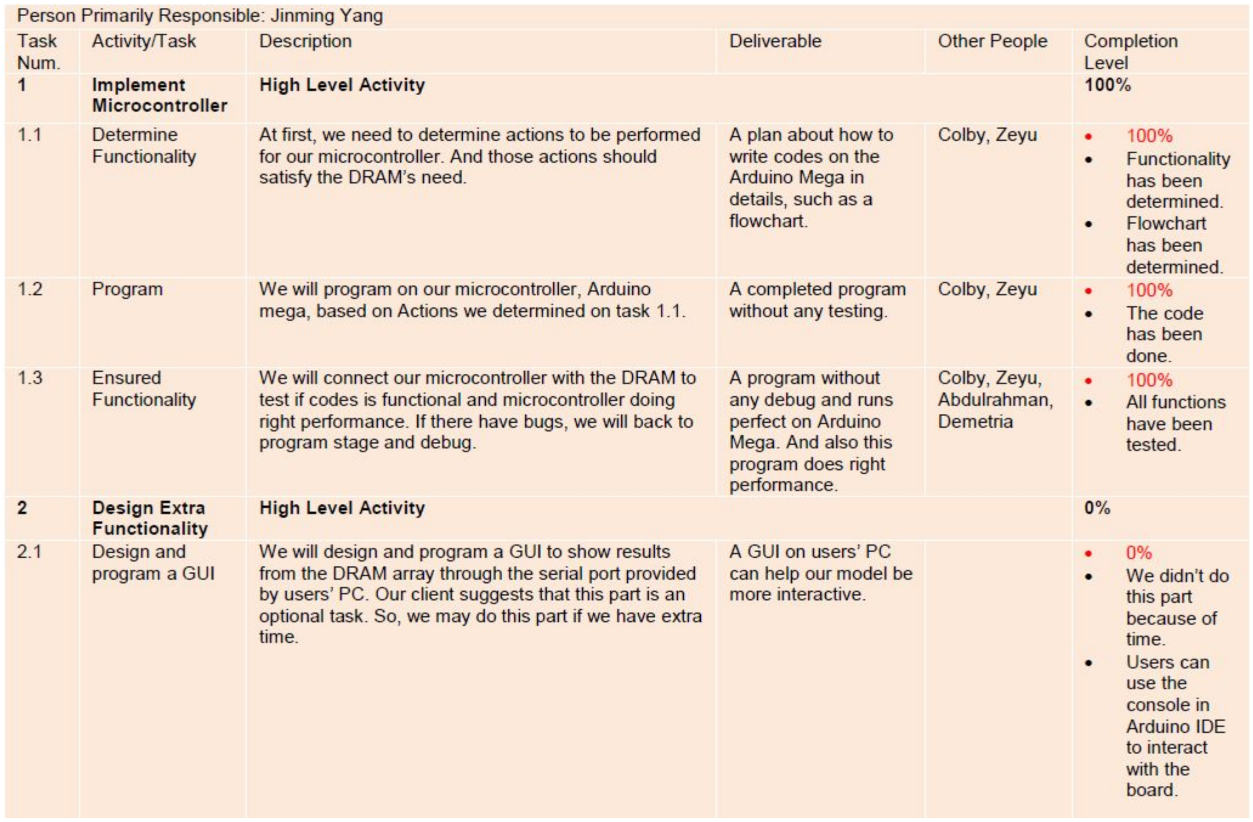
Demetria's WBS



| | Activity/Task(s) | Description | Deliverables | Other Team Member(s) | Levels of Completion |
|---|---|---|---|---|---|
| **1** | **Design the 8x8 (64 mbit cells) DRAM Array** | High-Level Activity<br>**Overall completion: 100%** | | | |
| 1.1 | Prototype/circuit design | The first stage of creating circuitry for the mbit cells. We will use the prototype design we chose as a team. | → Blueprint of the mbit cells location.<br>→ Schematic of the mbit cells. | All members | **100%**<br>→ Completed schematic on Fritzing software |
| 1.2 | Test on a temporary breadboard. | Test the first *four* digit- & word-lines for the first 16 mbit cells. | → Completed test the first *four* mbit cells followed by the digit- & word-lines. | All members | **100%**<br>→ Functionality working properly. |
| 1.3 | Design the entire 64 mbit cells on the white breadboards. | After prototype and circuit completion, build the entire 64 mbit cells on white breadboards. | → Have 4 of the 16 mbits done and put it together to make 64 mbits. | All members | **100%**<br>→ First 16 mbit cells completed, 3 more sets to go. |
| **2** | **Design Carrying Tray for the 8x8 DRAM Array** | Modern-Level Activity<br>**Overall Completion: 100%** | | | |
| 2.1 | Measure the full size of the DRAM array | Determine the size/dimension and material for the case. | → Sketch of the case design according from the dimension | All members. | **100%**<br>→ 20x7x6 inches overall size |
| 2.2 | Completed "tray" design for the DRAM Array | Ensure the "tray" is fitted securely for the final DRAM Array. | → Completed tray design. | All members | **100%**<br>→ Completed tray |

## Appendix E

### Jinming's WBS



Person Primarily Responsible: Jinming Yang

| Task Num. | Activity/Task | Description | Deliverable | Other People | Completion Level |
|---|---|---|---|---|---|
| **1** | **Implement Microcontroller** | **High Level Activity** | | | **100%** |
| 1.1 | Determine Functionality | At first, we need to determine actions to be performed for our microcontroller. And those actions should satisfy the DRAM's need. | A plan about how to write codes on the Arduino Mega in details, such as a flowchart. | Colby, Zeyu | • 100%<br>• Functionality has been determined.<br>• Flowchart has been determined. |
| 1.2 | Program | We will program on our microcontroller, Arduino mega, based on Actions we determined on task 1.1. | A completed program without any testing. | Colby, Zeyu | • 100%<br>• The code has been done. |
| 1.3 | Ensured Functionality | We will connect our microcontroller with the DRAM to test if codes is functional and microcontroller doing right performance. If there have bugs, we will back to program stage and debug. | A program without any debug and runs perfect on Arduino Mega. And also this program does right performance. | Colby, Zeyu, Abdulrahman, Demetria | • 100%<br>• All functions have been tested. |
| **2** | **Design Extra Functionality** | **High Level Activity** | | | **0%** |
| 2.1 | Design and program a GUI | We will design and program a GUI to show results from the DRAM array through the serial port provided by users' PC. Our client suggests that this part is an optional task. So, we may do this part if we have extra time. | A GUI on users' PC can help our model be more interactive. | | • 0%<br>• We didn't do this part because of time.<br>• Users can use the console in Arduino IDE to interact with the board. |

## Appendix F

### Zeyu's WBS

| **Person Primarily Responsible: Zeyu Zhang** | | | | | |
|---|---|---|---|---|---|
| **Task Num.** | **Activity/Task** | **Description** | **Deliverable** | **Other People** | **Level of Completion** |
| **1** | **Hardware (DRAM array)** | **This part is the core part of our project.** | **the completed DRAM array** | **All team** | **100%** |
| **1.1** | **Design DRAM cell** | **The basic structure in one cell is 1T1C and there should be an 8 by 8 (64 cells in total) array.** | **- - - -** | **Colby, Jinming** | **100%** |
| **1.2** | **Assemble DRAM array** | **After the DRAM cell passed all of requirement test, the DRAM array will be assembled by the 64 cells.** | **- - - - -** | **All team** | **100%** |
| **2** | **Software (Arduino)** | **Develop the controller code which could charge the DRAM array automatically.** | **the external controller (Arduino Mega) with implementation code** | **Colby, Jinming** | **100%** |
| **3** | **Robust Board** | **Build an unbreakable board to hold the DRAM array, Arduino and voltage source (battery).** | **the completed DRAM array with an external controller and robust board** | **All team** | **100%** |