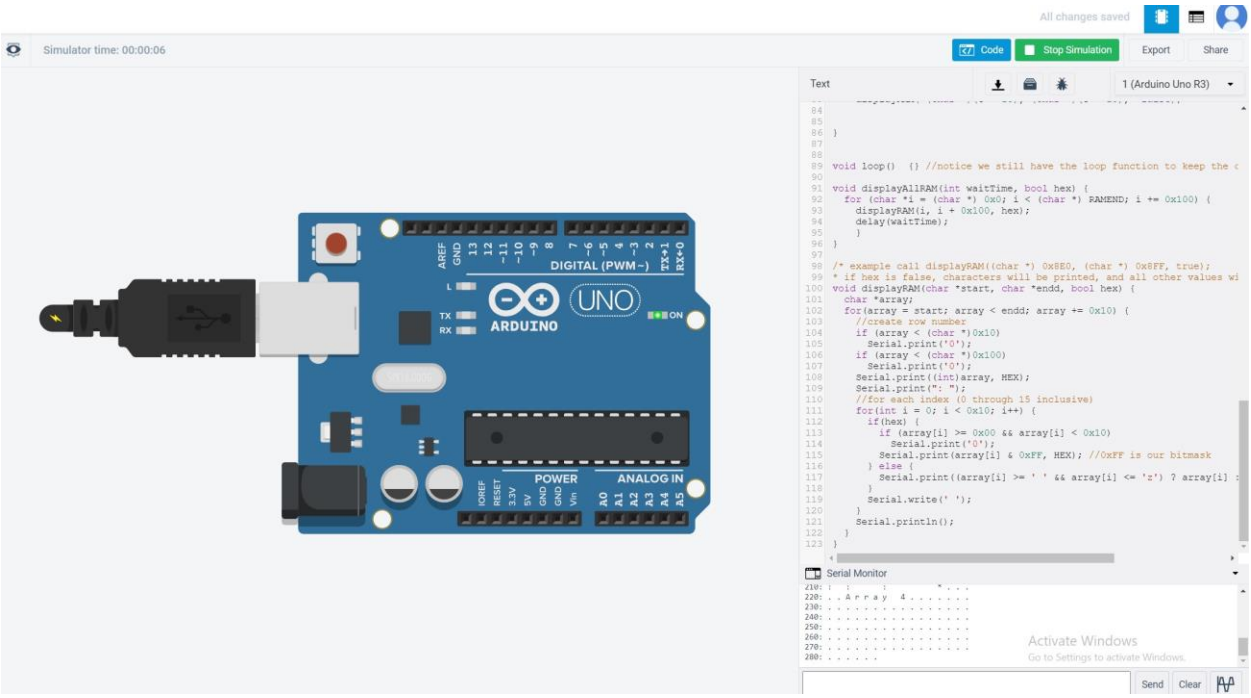
LAB REPORT #3

(Names) Kenilkumar Patel, Thieu Nguyen
(Date) 3/4/2021
Screenshot and components



EEPROM: the configuration data being transferred. Non - volatile.
Summary:
First I copy and paste the codes. This lab teaches the class how to write in the EEPROM bytes and store the data. EEPROM data is kept through power cycles and the stored data can be checked for corruption using an implemented checksum() method. First we will initialize some characters in the bytes and then check if the data has been corrupted through checksum. Then, we will check to see how EEPROM stores data that is two bytes or more. Result: initial check
000
001
002
003
004
005
006
007
008

EEPROM has been reinitialized

000 A
001 B
002 C
003 D
004 E
005 F
006 G
007 H
008 ÷

For part two of the lab, the new value at address 0x007 is ‘x’. This character shows up instead of 7 because the EEPROM uses an extended ASCII data table in order to make up for its storing and reading one byte. Normally, since Arduino is little Endian it would have print out 7 since 14200 in hex is 3778, storing 78 first then 37.

Conclusions:

I have learned that EEPROM only stores and reads one byte at a time. I also learned how to write and store data at specific bytes using the EEPROM library. This lab also helped me understand that Arduino is little endian. I have tried storing multiple characters at one byte and the EEPROM would only write out the last character. I also tried storing numbers but EEPROM only returned characters as well. #include <EEPROM.h>

```
void setup()
{
  Serial.begin(9600);

  // for test purposes only
  // EEPROM.write(0, 'h');  // overwrite something to simulate data corruption

  // print initial state of EEPROM
  Serial.println("initial check");
  printEEPROM();

  // if checksum is not OK, reinitialize EEPROM to default values
  if((checkSum() & 0xff) != EEPROM.read(0x08)) {
          initialize();
    Serial.println("EEPROM has been reinitialized");
  }
  else
  {
          Serial.println("EEPROM checksum is OK");
  }

  printEEPROM();
```

```
}
void loop() {}

void printEEPROM() {
  for(int address = 0; address < 0x09; address++) { // read
    a byte from the current address of the EEPROM char
    value = EEPROM.read(address);

    if (address < 0x10)
      Serial.print("00"); else
    if (address < 0x100)
    Serial.print("0");
    Serial.print(address, HEX);
    Serial.print("\t");
    Serial.print(value);
    Serial.println();
  }
  Serial.println();
}

void initialize() {
          //put some initial values in addresses 0x0 through 0x7
    //Example: EEPROM.write(0, 'B'); //store character B at address 0x0
    int twobytes = 14200; // For part 2
          EEPROM.write(0, 'A');
          EEPROM.write(1, 'B');
          EEPROM.write(2, 'C');
          EEPROM.write(3, 'D');
          EEPROM.write(4, 'E');
          EEPROM.write(5, 'F');
          EEPROM.write(6, 'G');
          EEPROM.write(7, 'H');



       //For part 2
     // EEPROM.write(7, twobytes);

    EEPROM.write(8, checkSum()); //recalculate checksum and store it at 0x8
          //Serial.println(twobytes, HEX);

}
```

```
char checkSum() {
  //implement a checksum
  algorithm char sum; byte a =
  EEPROM.read(0);
  byte b = EEPROM.read(1); byte
  c = EEPROM.read(2); byte d =
  EEPROM.read(3);
  byte e =
  EEPROM.read(4); byte f =
  EEPROM.read(5); byte g
  = EEPROM.read(6); byte
  h = EEPROM.read(7);
  sum = a ^ b; sum = sum ^
  c; sum = sum ^d; sum =
  sum ^e; sum = sum ^ f;
  sum = sum ^ g; sum =
  sum ^ h;

  return ~sum; //just a default value so the code compiles
}
```