With Support from National
Science Foundation &
National Security Agency
(Award# H98230-22-1-00098)

GenCyber 2023 Tennessee Tech

Arduino Password Gen

Student's Guide

Arduino Uno Random Password Generator

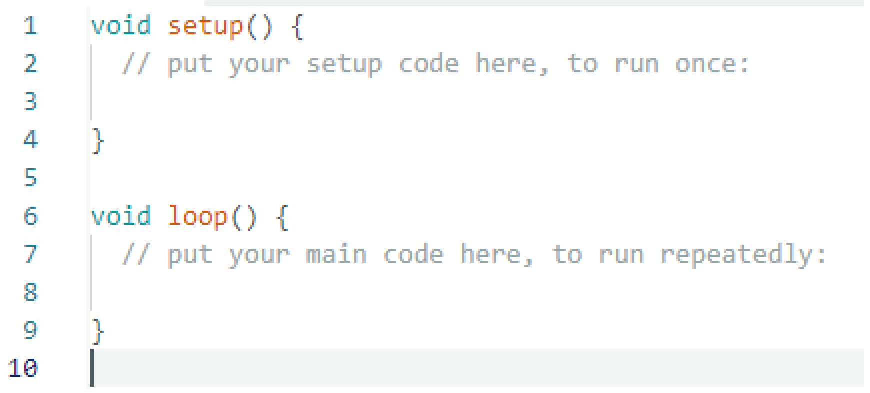
- Overview -

What is an Arduino Uno?

An Arduino Uno is a microcontroller board. A microcontroller is an electronic component that a single set of instructions repeatedly. These boards are commonly imbedded into larger boards to perform more complicated tasks. With boards like the Arduino Uno, you can write a program that grabs data from sensors or other types of input and perform tasks based off the input. In this activity you will be programming an Arduino Uno to generate a password when you press a button.

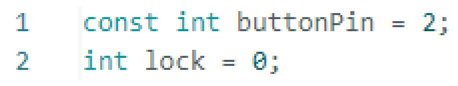
Step 01: Programming

The first step of any Arduino project is to lay out what you want to do and then program the board to fulfil that intended purpose. In this exercise, we will be making a program that will provide a secure password when you push a button. If you have programmed before, these programs look a lot like C/C++ programming. So, go to the Arduino IDE software and open an empty file. The first thing all programs need is the init() and loop() functions pictured below.



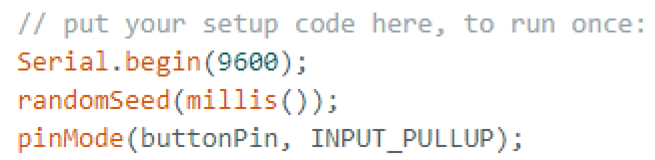
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}

```

As you might suspect, the setup function will only run once at the start of the program so this function is often filled with setup before actual work is done. The loop function will continually run until the program is stopped and this is where the actual work will be done. Before we move on to what is inside those function we will need two global variables at the top of the file as seen below.



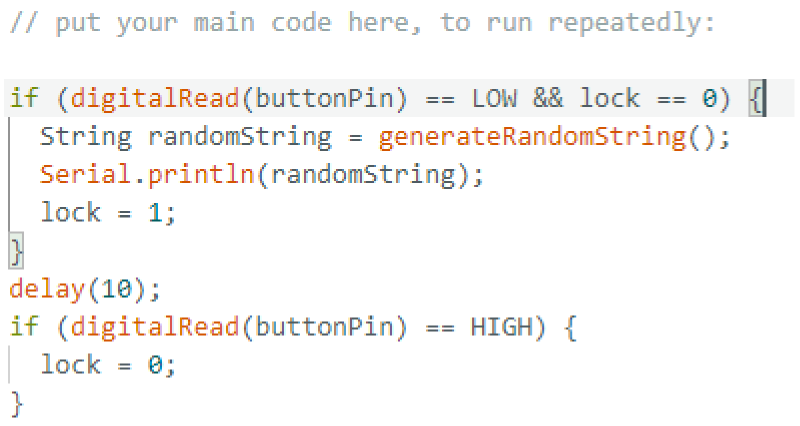
```
const int buttonPin = 2;
int lock = 0;
```

These line assign the value of 2 to buttonPin and 0 to lock. Lock will be explained later but buttonPin is the variable that describes where on the board to look for the button input, so when we get to the hardware portion we will be using pin 2.



```
// put your setup code here, to run once:
Serial.begin(9600);
randomSeed(millis());
pinMode(buttonPin, INPUT_PULLUP);
```

This is what should be inside of the setup() function. The first line *Serial.begin(9600)* sets up a connection between the board and your computer on the frequency of 9600. The second line *ranndomSeed(milis());* sets up a random number generator based on the current millisecond. The last line *pinMode(buttonPin, INPUT_PULLUP);* sets up the connection to the button as an input on pin 2 and sets a standard value so that it can know when the button is pressed.

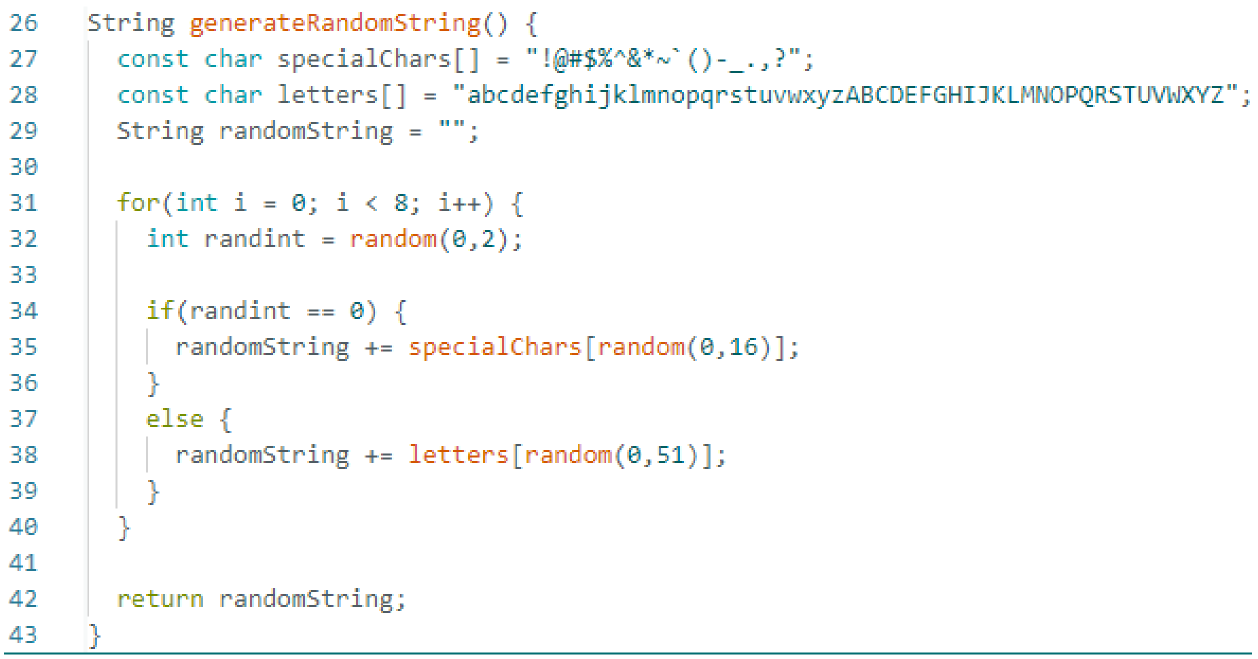


```
// put your main code here, to run repeatedly:

if (digitalRead(buttonPin) == LOW && lock == 0) {
  String randomString = generateRandomString();
  Serial.println(randomString);
  lock = 1;
}
delay(10);
if (digitalRead(buttonPin) == HIGH) {
  lock = 0;
}
```

This is a lot so let's start with the first if statement. It checks two things, first it reads the input value of pin 2 and compares it to LOW (checks if the button is pressed) and the second compares lock to 0. If both of those things are true (&&) then the next three lines are executed. So randomString will be generated by the generateRandomString() function which we will make later. *Serial.println(randomString);* will print the randomly generated string to the link to your computer so you can see it in the Arduino IDE. Lock will be set to 1.

After the first if statement there is a delay of 10 seconds. Then the second if statement is hit. This statement just checks the value of pin 2 and compares it to HIGH (checks if the button is not pressed). If that is true then it will set lock back to 0. If we did not have this if statement then the lock would not be reset and we would only get one good button press/password generated. The lock restricts the password generation to one password per push of the button, without the lock, as long as the button is pressed passwords will be continually generated.

The final step of the program is to define the function that randomly generates the password which is pictured below.



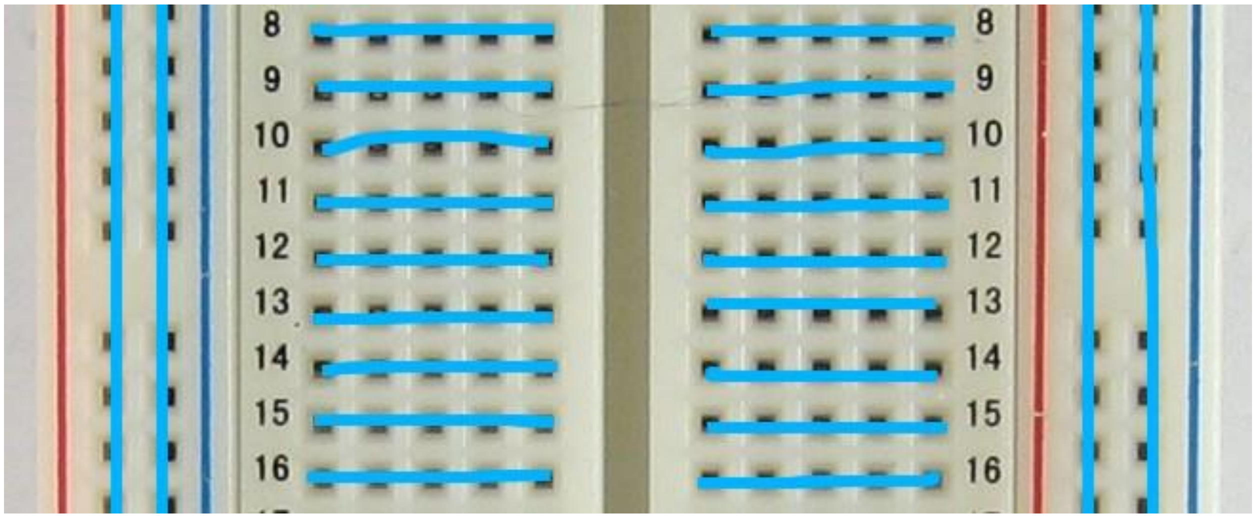
```
String generateRandomString() {
  const char specialChars[] = "!@#$%^&*~`()-_.,?";
  const char letters[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
  String randomString = "";

  for(int i = 0; i < 8; i++) {
    int randint = random(0,2);

    if(randint == 0) {
      randomString += specialChars[random(0,16)];
    }
    else {
      randomString += letters[random(0,51)];
    }
  }

  return randomString;
}
```

The first line says the name of the function and that it will produce a string value. The next two lines define two arrays that outline all the possible special characters and letters that can be used when generating the string. The next line *String randomString* = *“”;* defines the variable that we will be building the string in.

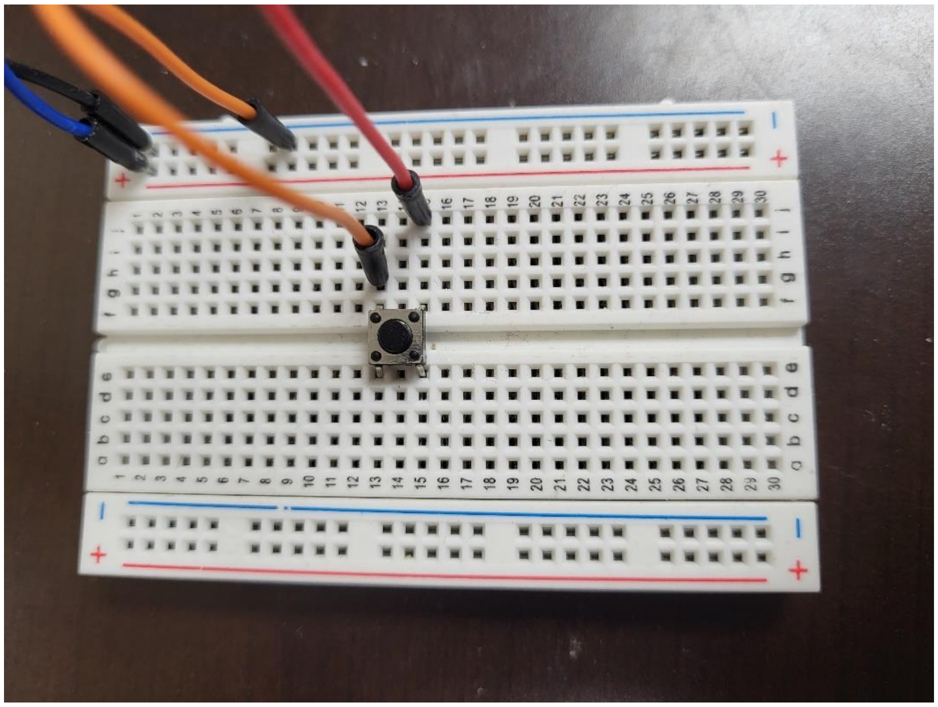
The next section is a for loop, as we have defined, it will loop 8 times (0-7). So for each run of the loop first it will generate a 0, 1, or 2 and assign the value to *randint*. If *randint* is 0 then it will randomly pick a special character and add it to the end of *randomString*. If *randint* is 1 or 2 then it will randomly pick a normal letter and add it to the end of *randomString*. Once the loop has run 8 times then the function will return the *randomString* it has built.

Step 02: Hardware

Go ahead and plug the USB cable into the Arduino Uno and a port on the computer. Also take out a button, 4 wires, and the breadboard. There are a couple things to note about breadboarding before we get started.

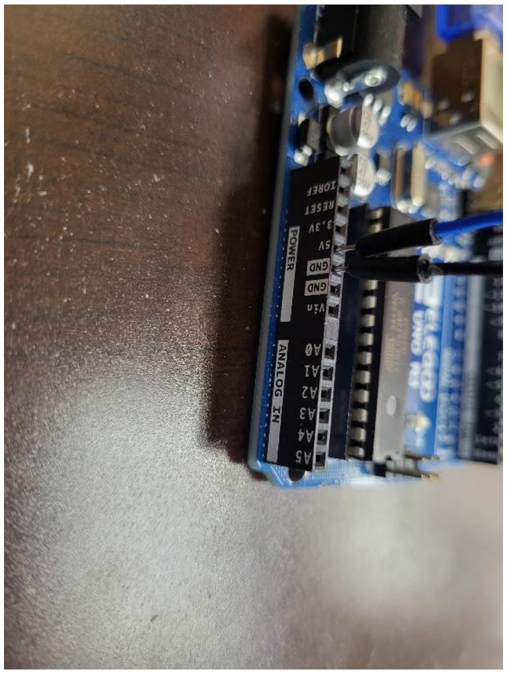


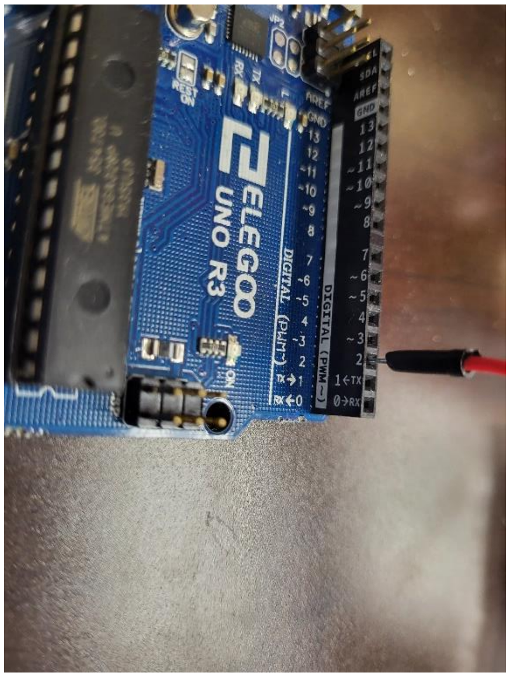
This image shows the various rails on the breadboard. Rails refer to the ports that are connected together, so when one port receives power all the other ports also receive power. So as the image shows, there are 2 long rails on each side of the breadboard. These specific rails are used to provide power (+) and ground (-) the board. For each row there are two sperate rails that are not connected together. When wiring projects please keep in mind where the power will actually flow.



Please set up your breadboard to look similar to the image above.

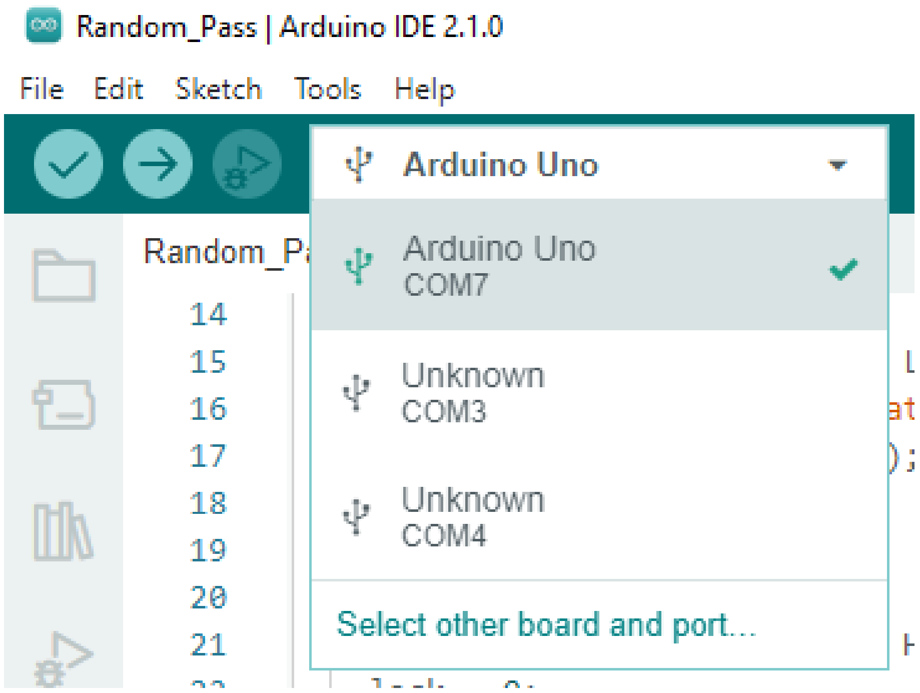
Have one wire in the top of both of the power and ground rails. Place the button to where it covers the divide between the two section of the row rails. Have one wire connect one side of the button to the ground rail and the final wire connects the other side of the button to the Arduino as pictured below. Also connect the power wire to the 5V port and the ground wire to one of the ports labeled GND on the Arduino as pictured below.



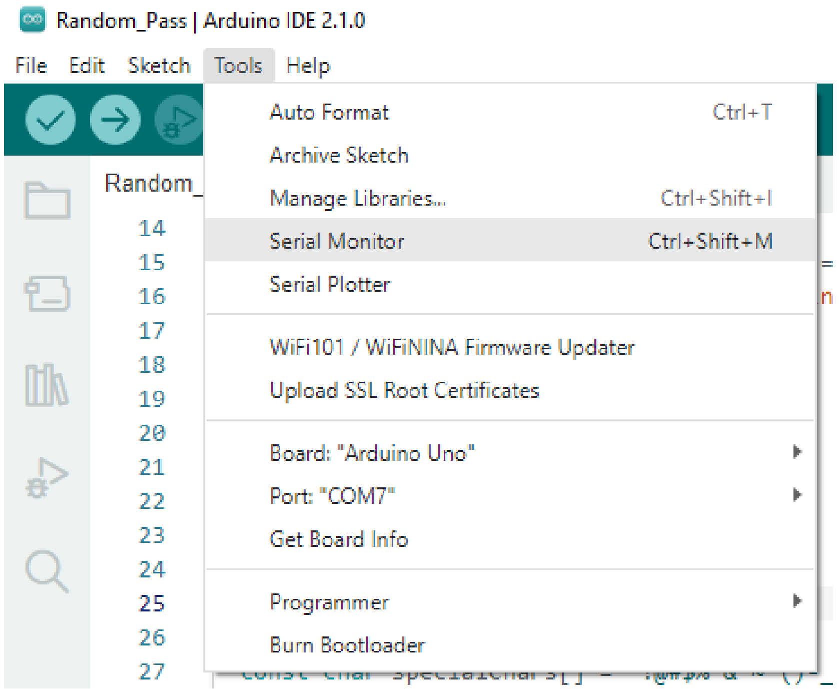


Step 03: Execution

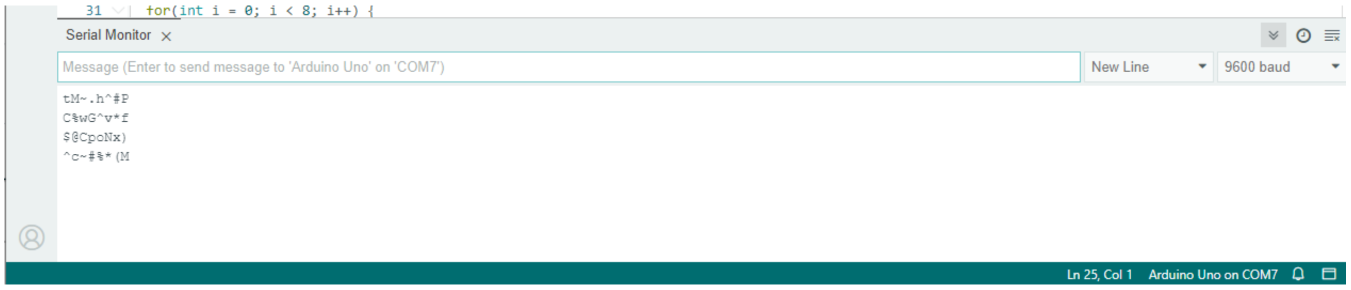
To execute the program, we first need to compile it. To do this just click the checkmark in the Arduino IDE, if there are any errors that appear please review the above section about software. Then we will need to upload the code to the Arduino board. First make sure the Arduino is plugged into the computer and then select the Arduino in the dropdown pictured below. Once that is selected you can click the right arrow to upload the code to the board.


Random_Pass | Arduino IDE 2.1.0

To see the password that is generated you will need to go under tools and click Serial Monitor and make sure the baud is the same as you set in the code for us it should be 9600.


Random_Pass | Arduino IDE 2.1.0

Then each time you push the button you should see a new password appear in the serial monitor window pictured below.



```
for(int i = 0; i < 8; i++) {
```

```
tM~.h^#P
C%wG^v*f
$@CpoNx)
^c~#%*(M
```

Step 04: Other Resources

This project was modified from this original creator: https://lilkittykat.hashnode.dev/a-password-generator-on-an-arduino

If you would like to download the Arduino IDE at home you can find it here: https://www.arduino.cc/en/software

For other types of projects, you can go here: https://docs.arduino.cc/built-in-examples/