Session 5: Pin Code Entry with Confirmation

TABLE OF CONTENTS

Introduction

In this session, we are going to review capturing a pin code and setting up a feedback mechanism to indicate whether we put in the right code or not. At this point, you should have the following code in your Keypad sketch to print out each key as it is pressed. If you don't, copy and paste the code below into your sketch.

C++

```
#include <Keypad.h>

const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {11, 10, 9, 8};
byte colPins[COLS] = {7, 6, 5, 4};

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

int digit = 1;

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

void loop(){
  char key = keypad.getKey();

  if (key != NO_KEY)
  {
    Serial.println(key);
  }
}
```

Capturing a Pin Code

Now that our keypad is working, we need to capture a pin code and verify it!

Here's how we're going to do it:

- Each time a button is pressed, record the key value by storing it in an array.
- Once we have received 4 values, check it against the correct pin code we store in code.

Types in C++

Before we dive into writing code, let's talk a little bit about types in C++.

When you create a variable in C++, you have to give it a type. Here are the types we will be using today:

int - integer - a positive or negative whole number. Range: **-32,000 to 32,000**
char - character - a letter or number in single quotes. 'A', 'Z', '5', '!'
byte - byte - whole number like integer but the range is: **0 to 255**
Any of these types can be defined as a constant (const) if the value does not need to change while the program is running.

Pin Code Length

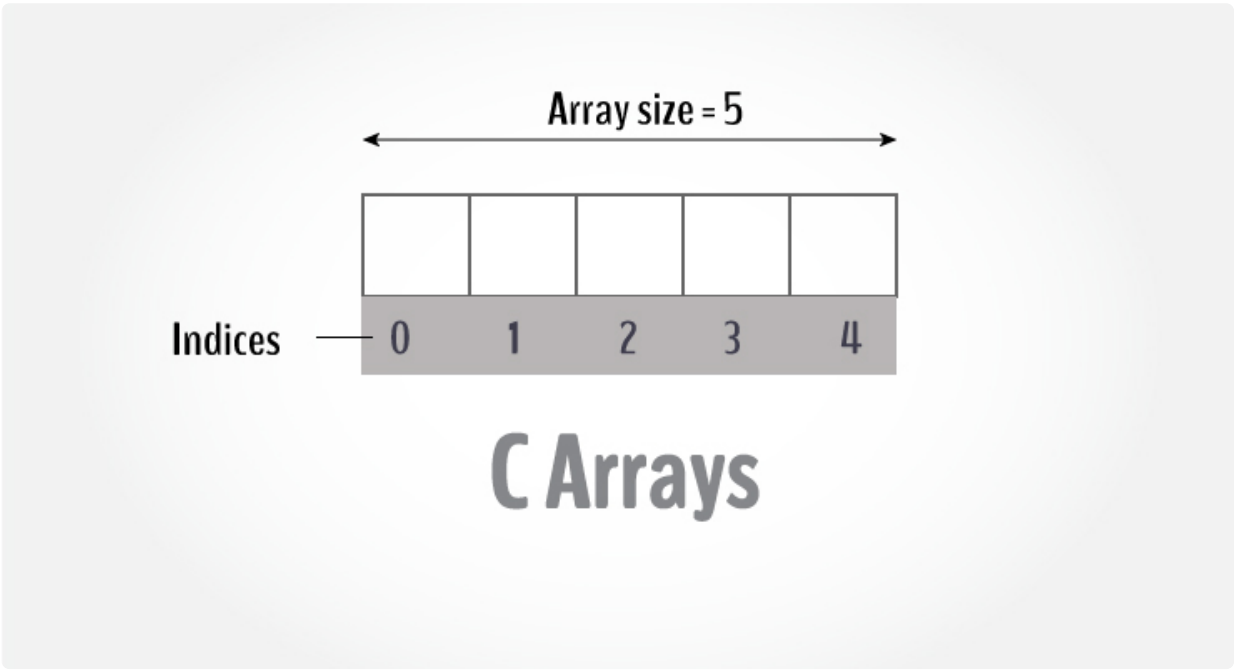
When programming, it's a good idea to start by defining your constants. First, let's define how long we want our pin code to be. Let's start with 4 digits. Make a constant **int** and set it to 4.

```
const int PIN_CODE_LENGTH = 4; //Length of our pin code
```

Next, let's set the correct pin code by defining a constant **char** array and setting it to whatever we want the correct pin code to be.

```
const char correctPinCode[PIN_CODE_LENGTH] = {'1','2','3','4'};
```

What is an array?
An array is basically a list of numbers or characters. When we are talking about a number or character in an array, we call it an **element**. We call the position each element is stored at the **index**. One quirky thing about arrays is the first element sits at index 0.



Correct Pin Array

When we define an array, if we know what goes in it, we can set up the elements right away. Here's the syntax to do that.

C++

```
const char correctPinCode[PIN_CODE_LENGTH] = {'1','2','3','4'};
```

Don't forget to add the const since it will not be changing while we run the program. We always need to define the length of the array so that the program knows how long to make our array.

Input Pin Array

Next we will create the array to store the characters we receive when the keypad buttons are pressed.

C++

```
char pinCodeInput[PIN_CODE_LENGTH];
```

We won't set it to any specific value like we did the correct pin code array because we want to fill it up as we receive characters.

Input Digit Tracker

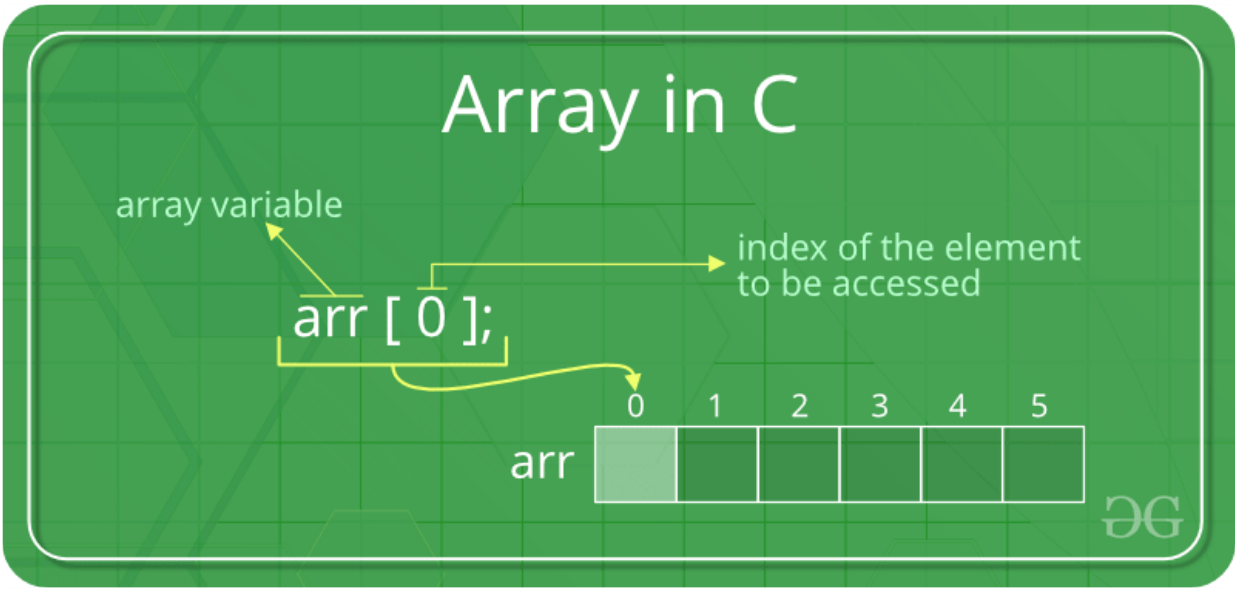
We will also need a variable that keeps track of which digit of the pin code we are about to record. We can make that an int and set it to 1, since we are starting with the first digit.

Now all our variables are set up. How will we capture each character and write it to the array?

Capturing Keypad Characters

```
16 void setup(){
17   Serial.begin(9600);
18 }
19
20 void loop(){
21   char key = keypad.getKey(
22
23   if (key != NO_KEY){
24     Serial.println(key);
25   }
26 }
```

See the **if** statement where we are printing the keypad characters we receive? We only get inside of this code block when the program has detected a keypad button press. It then prints out the character it receives. We can see above that key is defined as a **char**. This is why we made our arrays char type. Here is where we can capture that character and put it into our input array. Here's how to do that:



C++

```
void loop(){
  char key = keypad.getKey();

  if (key != NO_KEY){
    Serial.println(key);
    pinCodeInput[digitTracker - 1] = key;
  }
}
```

Great! Now we need to check if we just wrote the fourth digit. Use an **if statement and don't forget that we always use TWO equals signs in if statements.**

C++

```
void loop(){
  char key = keypad.getKey();

  if (key != NO_KEY){
    Serial.println(key);
    pinCodeInput[digitTracker - 1] = key;
    if(digitTracker == 4)
    {
        //If our program makes it here, we just wrote the fourth digit
    }
  }
}
```

Now, we just got our fourth digit. We know our array is full of the 4 characters that were pressed. If we just pressed 1234 then our input array variable would be equal to {'1', '2', '3', 4'}.

Checking input against the correct pin code

So we now need to check if our input array is equal to the correct pin code array. To do that, we are going to use a function that is included in the **standard library.** That's a library that you don't have to include to use in Arduino.

The function we will use is **strncmp()**. You can google "c++ strncmp" to find the documentation for this function. The documentation will always tell you the return type and parameters for the function.

strncmp takes three paramters:
1) a character array
2) another character array
3) how many characters to compare

strncmp returns 0 (zero) if the character arrays have the same contents for the first number of characters specified, so we can check if the function returns zero by writing it like this:

C++

```
void loop(){
  char key = keypad.getKey();

  if (key != NO_KEY){
    Serial.println(key);
    pinCodeInput[digit - 1] = key;
    if(digit == 4)
    {
        //If our program makes it here, we just wrote the fourth digit
        if(0 == strncmp(inputPinCode, correctPinCode, PIN_CODE_LENGTH))
        {
          //If our program makes it here, the correct code was entered!
          //Let's print something so that we know we made it here!
          Serial.println("That's the correct pin code!");
        }
        else
        {
          //If our program makes it here, the wrong code was entered.
          //Let's print something so that we know we made it here!
          Serial.println("That is not the correct pin code.");
```

```
        }
      }
    }
}
```

Adjust our digit tracker

One last thing we need to do adjust our digit tracker so that when we receive the next character, it is written to the right spot in the input array. We also need to reset the tracker to 1 when we are ready to try a new pin.

C++

```
void loop(){
  char key = keypad.getKey();

  if (key != NO_KEY){
    Serial.println(key);
    pinCodeInput[digit - 1] = key;
    if(digit == PIN_CODE_LENGTH)
    {
        //If our program makes it here, we just wrote the fourth digit
        if(0 == strncmp(inputPinCode, correctPinCode, PIN_CODE_LENGTH))
        {
          //If our program makes it here, the correct code was entered!
          //Let's print something so that we know we made it here!
          Serial.println("That's the correct pin code!");
        }
        else
        {
          //If our program makes it here, the wrong code was entered.
          //Let's print something so that we know we made it here!
          Serial.println("That is not the correct pin code.");
        }
```

```
22            digit = 1;
23        }
24        digit = digit + 1;
25      }
26    }
```

Knowledge Check - Kahoot

Input the code on the facilitator's shared screen

Breakout Room 1

By the end of this breakout room, students should have their pin codes successfully checked against the correct pin code and printing a statement saying whether or not the pin code was correct.

Extras - Do these if you finish early

1) How do we change the correct pin code? Do it in your Tinkercad circuit and verify that it works.
2) Try modifying your code to check for a 5 digit pin code. Verify that it works in Tinkercad.
3) What would we need to change in the code if we wanted to check the pin code only when you press the '*' key?

Pin Code Audible/Visual Confirmation

Now that we have the program printing out when we have a valid pin code, let's add some sort of confirmation in hardware. Choose one of the options below:

Audio Confirmation

Use a piezo buzzer to make a beep when the correct pin code is input and a different beep when the incorrect pin code is input.

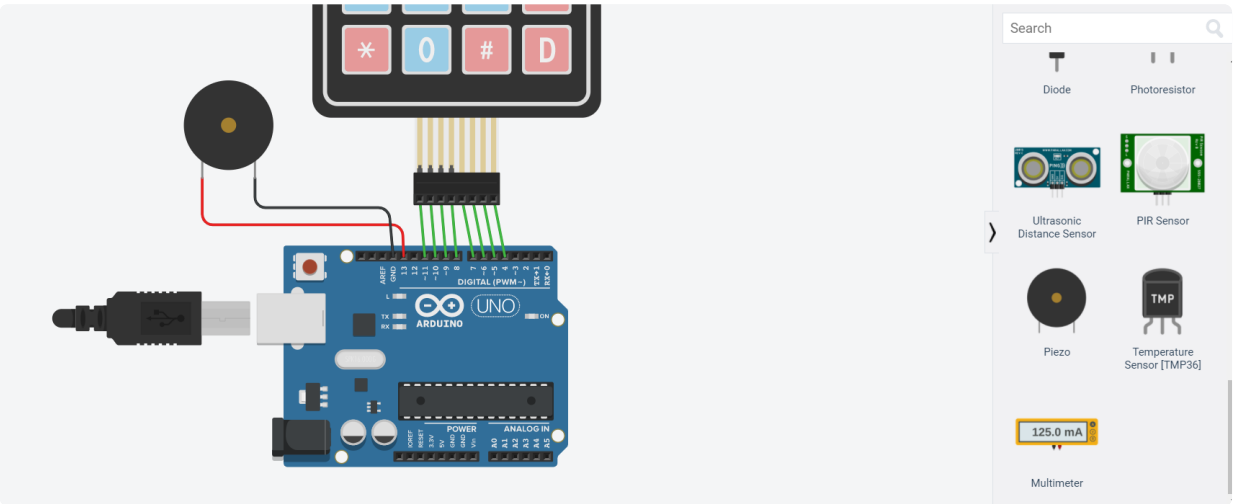
How-To

1) Open a **new circuit** and search for piezo in the component tray. Click on the component named **tone melody** under Starters and place it in your blank circuit. Click start simulation to see how it works. Click on the code button to show the code. Look at what kind of

variables are instantiated and where the majority of the code is. We are going to use part of this code to make our keypad sounds.
2) Visit the following link to find the code for the example circuit:

Arduino Tone Example Code

3)Find the piezo component from the component tray and add it to your keypad sketch. Connect the positive lead to pin 13 and the negative lead to the GND pin as shown below.



4) Look at the code and take out only the bits we need, adding them to the existing keypad code where it makes sense.

C++

```
#define NOTE_C4  262
#define NOTE_G3  196
#define NOTE_A3  22
#define NOTE_B3  247
```

These are the note definitions. You should put these at the top of your file, just below where you have #include <Keypad.h>

C++

```
// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3,
NOTE_C4
};

```

```
6   // note durations: 4 = quarter note, 8 = eighth note, etc.:
7   int noteDurations[] = {
8     4, 8, 8, 4, 4, 4, 4, 4
9   };
```

These are the two arrays that define how our melody will sound. We can modify the top one to change the notes we play and the bottom one to change how long those notes are played. We want to make two copies of each of these. One for when we get the right pin, and a different one for when we get the wrong pin. Name the arrays appropriately (something like correctPinMelody and wrongPinMelody, correctNoteDurations and wrongNoteDurations. You should put these below where our arrays are defined for the keypad.

C++

```
1   // iterate over the notes of the melody:
2     for (int thisNote = 0; thisNote < 8; thisNote++) {
3
4       // to calculate the note duration, take one second
5       // divided by the note type.
6       //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
7       int noteDuration = 1000 / noteDurations[thisNote]; //Change
    noteDurations to the name of the correct note durations array!
8       tone(13, melody[thisNote], noteDuration); //Change melody to
    the name of the correct melody array!
9
10      // to distinguish the notes, set a minimum time between them.
11      // the note's duration + 30% seems to work well:
12      int pauseBetweenNotes = noteDuration * 1.30;
13      delay(pauseBetweenNotes);
14      // stop the tone playing:
15      noTone(13);
16    }
```

This is the loop that runs through the arrays we have above to play the melody. You need to put one of these right below the serial print line for the correct pin code and one right below the serial print line for the incorrect pin code. Make sure the arrays referenced in each loop match the melody you want to play. If you changed how many notes are in your melody, change the number at the top of the for loop, where it says **thisNote < 8** to however long your melody is.

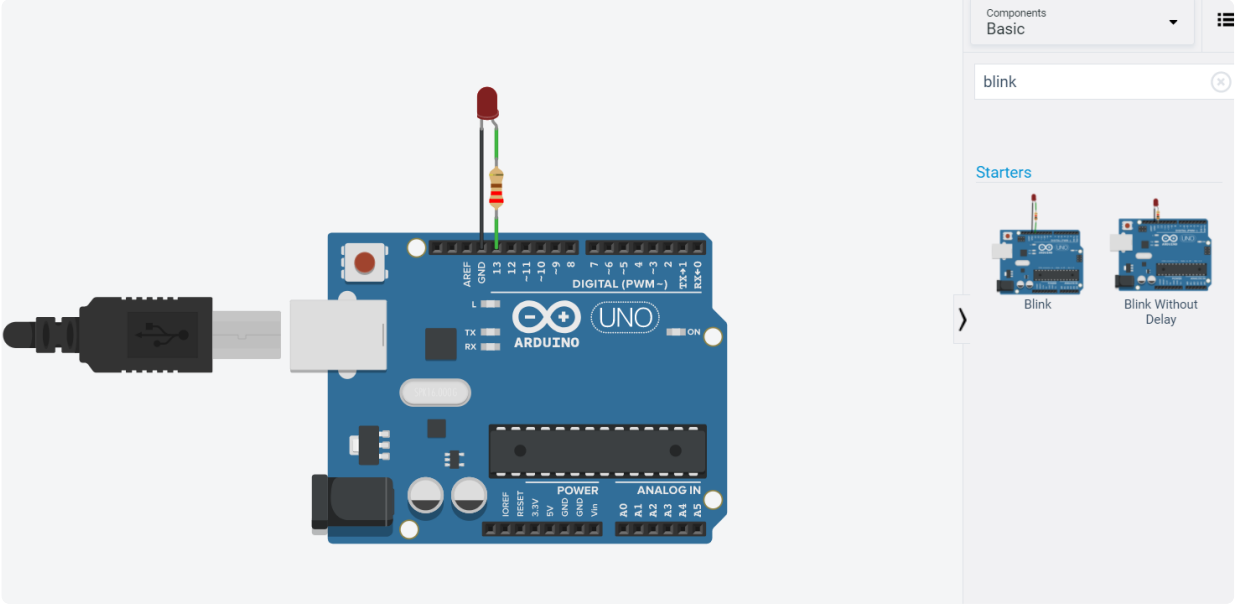
That's everything we need to generate a sound when we get the correct pin and a different sound when we get the incorrect pin code. Try your code out by simulating and trying a correct pin and incorrect pin. Open the serial monitor to help you determine what digit you are on.

Visual Confirmation

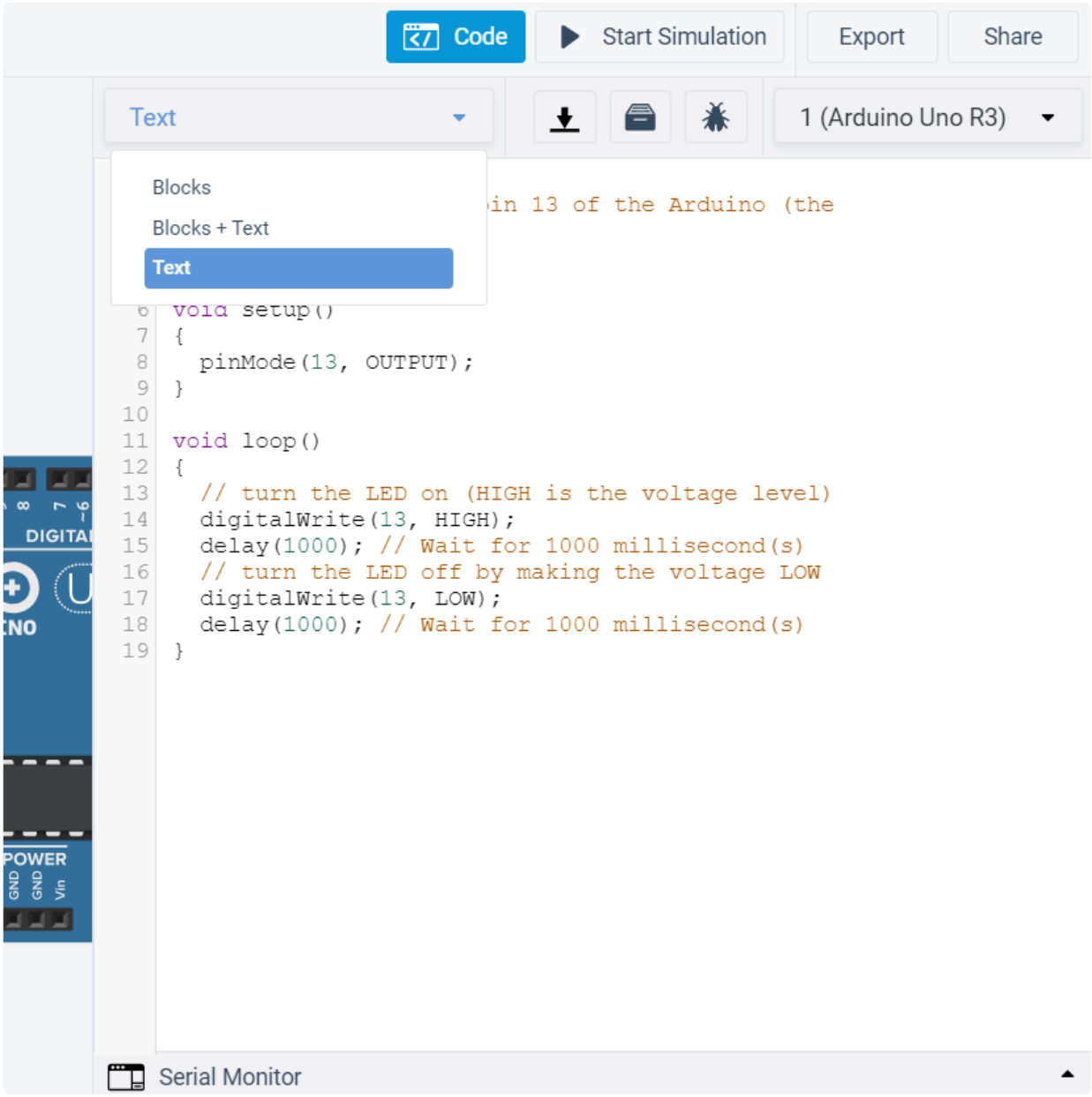
Use an RGB LED to indicate green when the correct pin code is input and red when the wrong pin code is input.

How-To

1) An RGB LED is just like the LEDs we worked with earlier this year except it has 3 colors in one LED. If you need to refresh yourself on how to turn on and off an LED, open a **new circuit** and search for the blink starter.

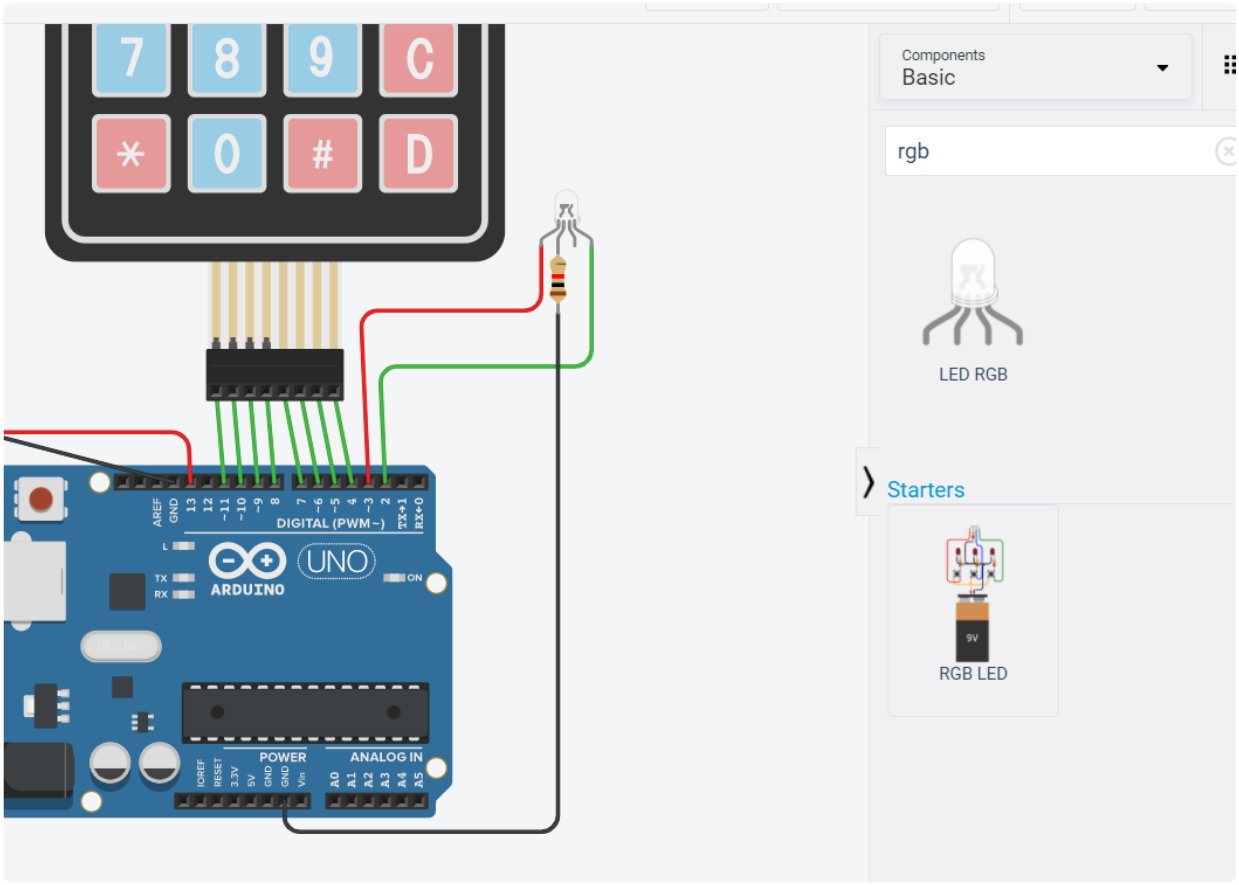


2) Place the component, open the code window, and switch the view to text.



Notice the line in the **setup()** function that sets the pin mode. We will need to do that for each pin we connect our RGB LED to.

3) Add an RGB LED to your keypad circuit. Find it by searching RGB in the component tray. Connect the cathode to ground with a 1K resistor and wire. Connect the red lead to pin 3 and the green lead to pin 2. You can change the color of the wire to help you remember which is hooked up to which pin



4) Add the code to set up pin 2 and pin 3 as output in the **setup()** function

C++

```
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
```

5) Add the code to turn on the green LED for 2 seconds after the serial print line when we get the correct pin code.

C++

```
// turn the LED on (HIGH is the voltage level)
digitalWrite(2, HIGH);
delay(2000); // Wait for 2000 millisecond(s)
// turn the LED off by making the voltage LOW
digitalWrite(2, LOW);
```

6) Add the code to blink the red LED twice after the serial print line when we get the wrong pin code.

C++

```
// turn the LED on (HIGH is the voltage level)
digitalWrite(3, HIGH);
delay(500); // Wait for 500 millisecond(s)
// turn the LED off by making the voltage LOW
digitalWrite(2, LOW);
delay(500); // Wait for 500 millisecond(s)
// turn the LED on (HIGH is the voltage level)
digitalWrite(3, HIGH);
delay(500); // Wait for 500 millisecond(s)
// turn the LED off by making the voltage LOW
digitalWrite(2, LOW);
```

That's it! Try your code out by simulating and trying a correct pin and incorrect pin. Open the serial monitor to help you determine what digit you are on.