ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Output 2: Online Course for Microcontrollers: syllabus, open educational resources

Practice leaflet: Module_1-1 pins as outputs

Lead Partner: International Hellenic University (IHU)

Authors: Theodosios Sapounidis [IHU], Aristotelis Kazakopoulos [IHU], Aggelos Giakoumis [IHU], Sokratis Tselegkaridis [IHU]

Erasmus+

GRANT NUMBER: 2020-1-PL01-KA226-HE-095653

Declaration

This report has been prepared in the context of the ENGINE project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

Funding Disclaimer

This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Table of Contents

Executive summary

In this Module we will use Arduino Uno pins as outputs to activate light or sound indicators.

Chapter 1: **Overview**

Table 1. Overview

Title / short summary	**Pins as outputs: Buzzer, LEDs and 7 segment display**
Expected learning outcomes	Students completing the course will be able to: • Recognize basic Arduino Uno functions and programming structures • Understand the define of pins as output • Design and implement simple circuits with LED, buzzer and 7 segment display
Keywords	Output pins, LED, Buzzer, 7 segment display
Duration	The duration of the module_1-1 is 3 hours • Module_1-1 slides - 30 minutes • 1st activity: Buzzer and LED - 50 minutes. Components are connected to the Arduino Uno to produce audio and visual alerts • 2nd activity: RGB LED - 50 minutes. Creating different colors through RGB LED • 3rd activity: 7 segment display - 50 minutes. Counter development showing from 0 to 9

Involved	**The students**: • Take part in activities • Complete code or circuit • Answer questionnaires **The teachers**: • Show the presentation of the module • Answer questions • Point out the tips • Encourage participation and discussion
Assignment	The module_1-1 includes: • 2 Open Projects
Educational tools and equipment	• Material: PC • Software: browser, Tinkercad
Prerequisites / pre-existing knowledge	• Students should have knowledge of wiring electronic components in breadboard (link1) • Students should have basic programming knowledge in C language (link2) • Students should be familiar with the Tinkercad environment (link3, tutorial video) • Students should have studied the educational material (slides) of Module_1-1

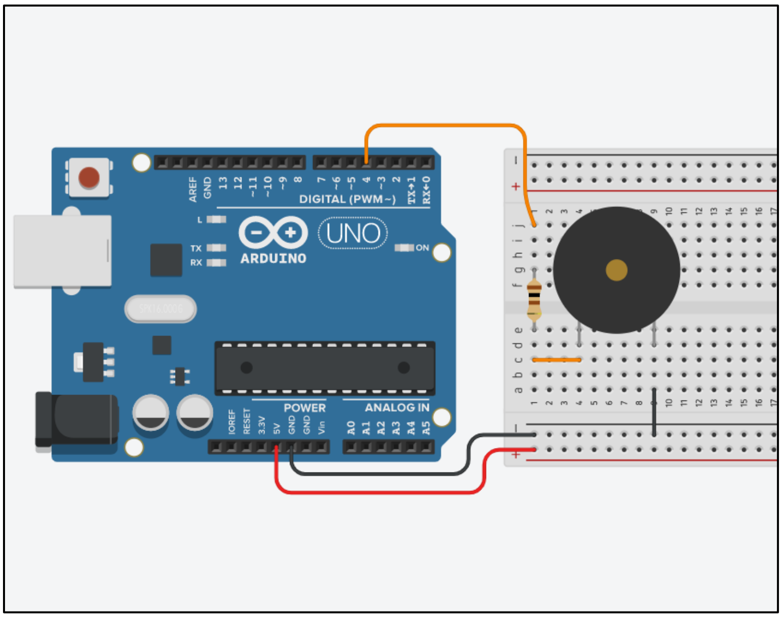
Educational content	Accompanying material: • Module_1-1 slides • Module_1-1 Evaluation leaflet • Module_1-1 Open Projects
Tips	***Tip1***. *Some components have polarity, and if connected incorrectly the circuit will not work* ***Tip2***. *Each program must include the setup () and loop () functions* ***Tip3***. *There are ON / OFF buzzer, but also buzzer that work with frequencies (link4)* ***Tip4***. *The RGB LED in Tinkercad is a common cathode* ***Tip5***. *A 7 segment display can be a common cathode or a common anode*

Chapter 2: **Activities**

2.1 Activity 1. Buzzer and LED

This activity utilizes Arduino Uno output pins to generate audio and / or visual alerts. The activity is divided into 3 parts: a) use of buzzer, b) use of LED, c) use of buzzer and LED.

Table 2. Activity 1

Activity 1a (15 minutes)	In this part the aim is to turn a buzzer on and off every 2.5 seconds. **Step 1**. Draw the circuit in Tinkercad. A buzzer is connected to the Arduino Uno **Step 2**. Write the microcontroller code **Step 3**. Simulate the circuit and test it
Step 1 (5 minutes)	Draw the next circuit in Tinkercad.  *Figure 1. Buzzer connection*

Step 2
(8 minutes)

Study the code and write it on the microcontroller:

```
/* Buzzer

Circuit Connections:
PIN_4  =>  Buzzer_Positive  -  Buzzer_Negative  =  >
Resistor 100Ω => Gnd
*/

//The  setup()  function  initializes  and  sets  the
initial values
//It will only run once after each powerup or reset
void setup()
{
  //Configures the PIN_4 to behave as output
  pinMode(4, OUTPUT);
}

//This function loops consecutively
void loop()
{
  digitalWrite(4, HIGH); //Write a HIGH value (5V)
to digital pin 4 - Buzzer on
  delay(2500);  //  Pauses  the  program  for  2500
milliseconds
  digitalWrite(4,  LOW);  //Write  a  LOW  value  (0V)
to digital pin 4 - Buzzer off
  delay(2500); // Wait for 2500 milliseconds
}
```

Step 3
(2 minutes)

Run the simulation and check the correct operation of the circuit

Activity 1b
(15 minutes)

In this part the aim is to turn on and off an LED every 1 second.

Step 1. Draw the circuit in Tinkercad. A LED is connected to the Arduino Uno

Step 2. Write the microcontroller code

Step 3. Simulate the circuit and test it

Step 1
(5 minutes)

Draw the next circuit in Tinkercad.

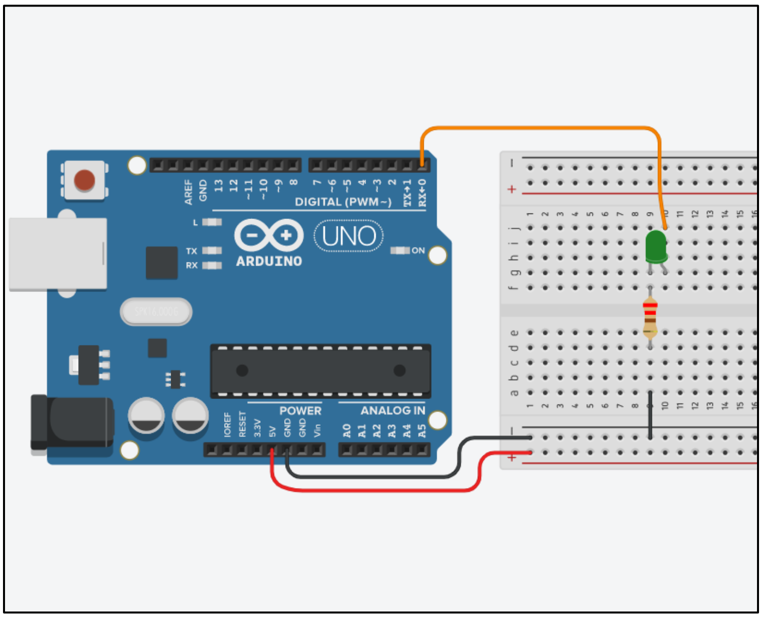
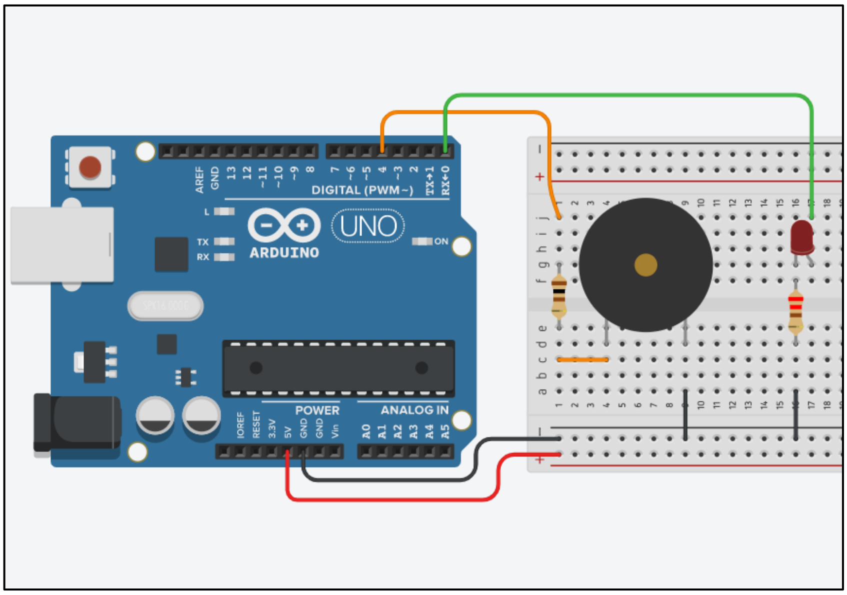


Figure 2. LED connection

Step 2
(8 minutes)

Study the code and write it on the microcontroller:

```
/* Blinking a LED

Circuit Connections:
PIN_0 => LED_Anode - LED_Cathode = > Resistor 220Ω
=> Gnd
*/

//The setup() function initializes and sets the
initial values
//It will only run once after each powerup or reset
void setup()
{
  //Configures the PIN_0 to behave as output
  pinMode(0, OUTPUT);
}

//This function loops consecutively
void loop()
{
  digitalWrite(0, HIGH); //Write a HIGH value (5V)
to digital pin 0 - LED on
  delay(1000); // Pauses the program for 1000
milliseconds
  digitalWrite(0, LOW); //Write a LOW value (0V)
to digital pin 0 - LED off
  delay(1000); // Wait for 1000 milliseconds
}
```

Step 3 (2 minutes)	Run the simulation and check the correct operation of the circuit
Activity 1c (20 minutes)	In this part the aim is to turn on and off alternately a buzzer and a LED every 1 second. **Step 1**. Draw the circuit in Tinkercad. A buzzer and an LED are connected to the Arduino Uno **Step 2**. Write the microcontroller code **Step 3**. Simulate the circuit and test it **Step 4**. Modifications and discussion
Step 1 (5 minutes)	Draw the next circuit in Tinkercad.  *Figure 3. LED and Buzzer connection*

Step 2
(8 minutes)

Study the code and write it on the microcontroller. The 2 missing lines must be completed:

```
/* LED and Buzzer

Circuit Connections:
PIN_0 => LED_Anode - LED_Cathode = > Resistor 220Ω
=> Gnd
PIN_4  =>  Buzzer_Positive  -  Buzzer_Negative  =  >
Resistor 100Ω => Gnd
*/

//The  setup()  function  initializes  and  sets  the
initial values
//It will only run once after each powerup or reset
void setup()
{
  //Configures the PIN_0 and the PIN_4 to behave
as outputs
  pinMode(0, OUTPUT);
=>                        //complete the line
}

//This function loops consecutively
void loop()
{
=>                         //Write a LOW value (0V)
to digital pin 0 – LED off
  digitalWrite(4, HIGH); //Write a HIGH value (5V)
to digital pin 4 – Buzzer on
  delay(2500);  //  Pauses  the  program  for  2500
milliseconds
  digitalWrite(0, HIGH); //Write a HIGH value (5V)
to digital pin 0 – LED on
  digitalWrite(4, LOW);  //Write a LOW value (0V)
to digital pin 4 – Buzzer off
  delay(2500); // Wait for 2500 milliseconds
}
```

Step 3
(2 minutes)

Run the simulation and check the correct operation of the circuit

Step 4
(5 minutes)

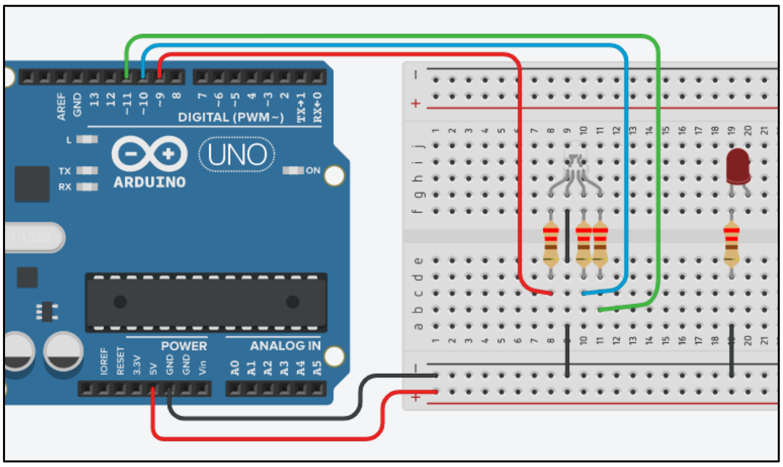
Suggested modifications and discussion:

- Replace the LED resistor with a new **10Ω resistor**. Run the simulation. What do you notice?
- Replace the LED resistor with a new **10KΩ resistor**. Run the simulation. What do you notice?
- Invert the LED. Run the simulation. What do you notice?

2.2 Activity 2. RGB LED

This activity utilizes Arduino Uno output pins to drive an RGB LED. The activity is divided into 2 parts: a) RGB LED vs LED, b) RGB LED.

Table 3. Activity 2

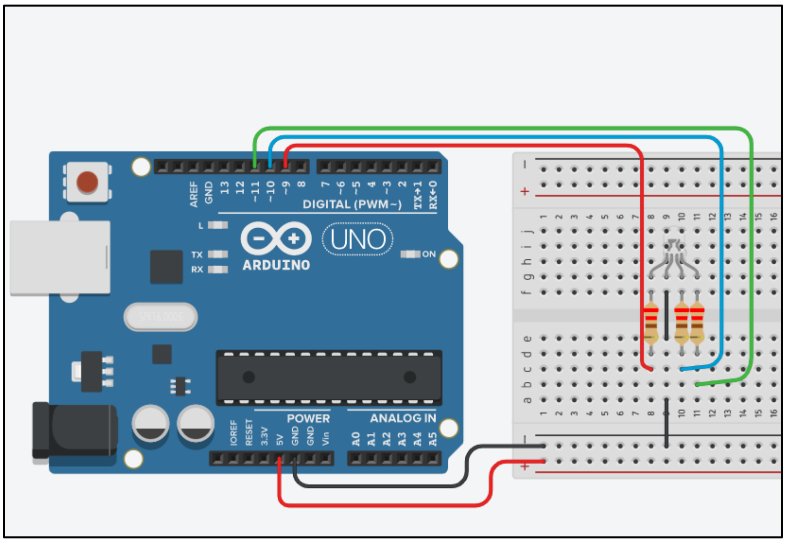
Activity 2a (25 minutes)	In this part the aim is to operate an RGB LED and compare it to a simple LED. More specifically, the RGB LED changes color between the three primary colors every second, while the simple LED lights up permanently. **Step 1**. Draw the circuit in Tinkercad. An RGB LED and a simple LED are connected to the Arduino Uno **Step 2**. Write the microcontroller code **Step 3**. Simulate the circuit and test it
Step 1 (8 minutes)	Draw the next circuit in Tinkercad and complete the LED connection so that its anode goes to pin 12 of the Arduino Uno.  AREF GND 13 12 ~11 ~10 ~9 8 7 ~6 ~5 4 ~3 2 TX→1 RX←0 DIGITAL (PWM~) L TX RX UNO ARDUINO ON POWER IOREF RESET 3.3V 5V GND GND Vin ANALOG IN A0 A1 A2 A3 A4 A5 *Figure 4. RGB LED and LED connection*

Step 2
(15 minutes)

Study the code and write it on the microcontroller:

```
/* RGB LED vs LED

Circuit Connections:
PIN_12 => LED_Anode - LED_Cathode = > Resistor 220Ω
=> Gnd
PIN_9   => Resistor 220Ω =>  Red pin of RGB LED
PIN_11 => Resistor 220Ω =>  Blue pin of RGB LED
PIN_10 => Resistor 220Ω =>  Green pin of RGB LED
*/

#define R_pin 9            //give the name “R_pin”
to PIN_9
#define G_pin 11           //give the name “G_pin”
to PIN_11
#define B_pin 10           //give the name “B_pin”
to PIN_10
#define LED_pin 12      //give the name “LED_pin”
to PIN_12

//The setup() function initializes and sets the
initial values
//It will only run once after each powerup or reset
void setup()
{
  //Configures the PIN_9, PIN_10, PIN_11 and
PIN_12 to behave as outputs
  pinMode(R_pin, OUTPUT);
  pinMode(G_pin, OUTPUT);
  pinMode(B_pin, OUTPUT);
  pinMode(LED_pin, OUTPUT);
}

//This function loops consecutively
void loop()
{
  digitalWrite(LED_pin, HIGH); //Write a HIGH
value (5V) to digital pin 12 – LED on

  //red color for RGB = > R=255, G=0, B=0
  analogWrite(R_pin, 255); //Write 100% PWM to pin
9
  analogWrite(G_pin, 0);  //Write 0% PWM to pin 11
  analogWrite(B_pin, 0);  //Write 0% PWM to pin 10
  delay(1000);            // Wait for 1 second

  //green color for RGB = > R=0, G=255, B=0
  analogWrite(R_pin, 0);   //Write 0% PWM to pin 9
  analogWrite(G_pin, 255); //Write 100% PWM to pin
11
  analogWrite(B_pin, 0);  //Write 0% PWM to pin 10
  delay(1000);            // Wait for 1 second

  //blue color = > RGB=0,0,255
  analogWrite(R_pin, 0);  //Write 0% PWM to pin 9
  analogWrite(G_pin, 0);  //Write 0% PWM to pin 11
  analogWrite(B_pin, 255); //Write 100% PWM to pin
10
```

	`  delay(1000);            // Wait for 1 second`
	`}`

Step 3 (2 minutes)	Run the simulation and check the correct operation of the circuit
Activity 2b (25 minutes)	In this part the aims is to operate an RGB LED, calling a function. Every 1 second the RGB LED changes color between: red, green, blue, magenta, yellow, white, silver, purple. **Step 1**. Draw the circuit in Tinkercad. An RGB LED is connected to the Arduino Uno **Step 2**. Write the microcontroller code **Step 3**. Simulate the circuit and test it **Step 4**. Modifications and discussion
Step 1 (8 minutes)	Draw the next circuit in Tinkercad.  DIGITAL (PWM ~) UNO ARDUINO POWER ANALOG IN AREF GND 13 12 ~11 ~10 ~9 8 7 ~6 ~5 4 ~3 2 TX→1 RX←0 L TX RX ON IOREF RESET 3.3V 5V GND GND Vin A0 A1 A2 A3 A4 A5 *Figure 5. RGB LED*

Step 2
(10 minutes)

Study the code and write it on the microcontroller:

```
/* RGB LED

Circuit Connections:
PIN_9   => Resistor 220Ω =>  Red pin of RGB LED
PIN_11 => Resistor 220Ω =>  Blue pin of RGB LED
PIN_10 => Resistor 220Ω =>  Green pin of RGB LED
*/

#define R_pin 9   //give the name “R_pin” to PIN_9
#define G_pin 11  //give the name “G_pin” to PIN_11
#define B_pin 10  //give the name “B_pin” to PIN_10

//The setup() function initializes and sets the
initial values
//It will only run once after each powerup or reset
void setup()
{
  //Configures the PIN_9, PIN_10, PIN_11 to behave
as outputs
  pinMode(R_pin, OUTPUT);
  pinMode(G_pin, OUTPUT);
  pinMode(B_pin, OUTPUT);
}

//This function loops consecutively
void loop()
{
  set_RGB(255, 0, 0);
  // call the function for the red color
  delay(1000);     // Wait for 1 second

  set_RGB(0, 255, 0);
  // call the function for the green color
  delay(1000);      // Wait for 1 second

  set_RGB(0, 0, 255);
  // call the function for the blue color
  delay(1000);      // Wait for 1 second

  set_RGB(255, 0, 255);
  // call the function for the magenta color
  delay(1000);          // Wait for 1 second

  set_RGB(255, 255, 0);
  // call the function for the yellow color
  delay(1000);          // Wait for 1 second

  set_RGB(255, 255, 255);
   // call the function for the white color
  delay(1000);      // Wait for 1 second

  set_RGB(192, 192, 192);
   // call the function for the silver color
  delay(1000);      // Wait for 1 second

  set_RGB(128, 0, 128);
   // call the function for the purple color
  delay(1000);       // Wait for 1 second
}

//This function set values in the RGB LED
void  set_RGB(int  R_value,  int  G_value,  int
B_value)
 {
  analogWrite(R_pin, R_value);
  //set a value (from 0 to 255) in PIN_9
  analogWrite(G_pin, G_value);
  //set a value (from 0 to 255) in PIN_11
  analogWrite(B_pin, B_value);
  //set a value (from 0 to 255) in PIN_10
}
```

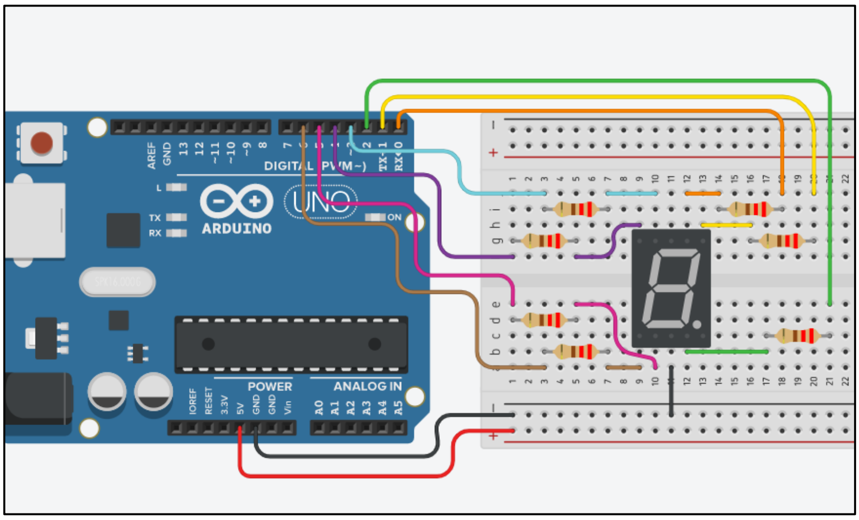
Step 3 (2 minutes)	Run the simulation and check the correct operation of the circuit
Step 4 (5 minutes)	Suggested modifications and discussion: • Can the RGB LED be connected to pins 0, 1, 2? Try it. What do you notice?

2.3 Activity 3. Seven segment display

This activity utilizes Arduino Uno output pins to drive a seven-segment display.

Table 4. Activity 3

Activity 3 (50 minutes)	The seven-segment display counts from 0 to 9, increasing the number every second. **Step 1**. Draw the circuit in Tinkercad. A seven-segment display is connected to the Arduino Uno **Step 2**. Write the microcontroller code **Step 3**. Simulate the circuit and test it **Step 4**. Modifications and discussion

Step 1 (15 minutes)	Draw the next circuit in Tinkercad.  *Figure 6. Seven segment display connection*

Step 2
(25 minutes)

Study the code and write it on the microcontroller:

```
/* Seven segment display

Circuit Connections:
Seven segment common Cathode = > Gnd
PIN_0 => Resistor 220Ω => Segment a
PIN_1 => Resistor 220Ω => Segment b
PIN_2 => Resistor 220Ω => Segment c
PIN_3 => Resistor 220Ω => Segment f
PIN_4 => Resistor 220Ω => Segment g
PIN_5 => Resistor 220Ω => Segment d
PIN_6 => Resistor 220Ω => Segment e
*/

#define A_pin 0 //give the name “A_pin” to PIN_0
#define B_pin 1 //give the name “B_pin” to PIN_1
#define C_pin 2 //give the name “C_pin” to PIN_2
#define D_pin 5 //give the name “D_pin” to PIN_5
#define E_pin 6 //give the name “E_pin” to PIN_6
#define F_pin 3 //give the name “F_pin” to PIN_3
#define G_pin 4 //give the name “G_pin” to PIN_4

//The  setup()  function  initializes  and  sets  the
initial values
//It will only run once after each powerup or reset
void setup() {
  pinMode(A_pin, OUTPUT);
  //Configure the PIN_0 to behave as output
  pinMode(B_pin, OUTPUT);
  //Configure the PIN_1 to behave as output
  pinMode(C_pin, OUTPUT);
  //Configure the PIN_2 to behave as output
  pinMode(D_pin, OUTPUT);
  //Configure the PIN_5 to behave as output
  pinMode(E_pin, OUTPUT);
   //Configure the PIN_6 to behave as output
  pinMode(F_pin, OUTPUT);
  //Configure the PIN_3 to behave as output
  pinMode(G_pin, OUTPUT);
  //Configure the PIN_4 to behave as output
}

//This function loops consecutively
void loop() {
  for (int i=0; i<10; i++){
    sevenSegment(i);
 //call the function and pass a number from 0 to 9
    delay(1000);
 //wait for 1 second
  }
}

//This  function  activates  and  deactivates  the
segments
//so the numbers appear on the display
void sevenSegment (int selection){
  switch(selection){
  case 0:
```

```
 /* display 0
       -
      |  |
      |  |
       -
   */
 digitalWrite(A_pin, HIGH);
 //activate segment A
 digitalWrite(B_pin, HIGH);
 //activate segment B
 digitalWrite(C_pin, HIGH);
 //activate segment C
 digitalWrite(D_pin, HIGH);
 //activate segment D
 digitalWrite(E_pin, HIGH);
 //activate segment E
 digitalWrite(F_pin, HIGH);
 //activate segment F
 digitalWrite(G_pin, LOW);
 //deactivate segment G
 break;

 case 1:
/* display 1

        |
        |

   */
 digitalWrite(A_pin, LOW);
 //deactivate segment A
 digitalWrite(B_pin, HIGH);
 //activate segment B
 digitalWrite(C_pin, HIGH);
 //activate segment C
 digitalWrite(D_pin, LOW);
 //deactivate segment D
 digitalWrite(E_pin, LOW);
 //deactivate segment E
 digitalWrite(F_pin, LOW);
 //deactivate segment F
 digitalWrite(G_pin, LOW);
 //deactivate segment G
 break;

 case 2:
 /* display 2
       -
        |
       -
      |
       -
   */
 digitalWrite(A_pin, HIGH);
 //activate segment A
 digitalWrite(B_pin, HIGH);
 //activate segment B
 digitalWrite(C_pin, LOW);
 //deactivate segment C
 digitalWrite(D_pin, HIGH);
 //activate segment D
 digitalWrite(E_pin, HIGH);
```

```
//activate segment E
digitalWrite(F_pin, LOW);
//deactivate segment F
digitalWrite(G_pin, HIGH);
//activate segment G
break;

case 3:
/* display 3
      -
       |
      -
       |
      -
  */
digitalWrite(A_pin, HIGH);
//activate segment A
digitalWrite(B_pin, HIGH);
//activate segment B
digitalWrite(C_pin, HIGH);
//activate segment C
digitalWrite(D_pin, HIGH);
//activate segment D
digitalWrite(E_pin, LOW);
//deactivate segment E
digitalWrite(F_pin, LOW);
//deactivate segment F
digitalWrite(G_pin, HIGH);
//activate segment G
break;

case 4:
/* display 4

     | |
      -
       |

  */
digitalWrite(A_pin, LOW);
//deactivate segment A
digitalWrite(B_pin, HIGH);
//activate segment B
digitalWrite(C_pin, HIGH);
//activate segment C
digitalWrite(D_pin, LOW);
//deactivate segment D
digitalWrite(E_pin, LOW);
//deactivate segment E
digitalWrite(F_pin, HIGH);
//activate segment F
digitalWrite(G_pin, HIGH);
//activate segment G
break;

case 5:
/* display 5
      -
     |
      -
       |
      -
  */
digitalWrite(A_pin, HIGH);
//activate segment A
digitalWrite(B_pin, LOW);
//deactivate segment B
digitalWrite(C_pin, HIGH);
//activate segment C
digitalWrite(D_pin, HIGH);
//activate segment D
digitalWrite(E_pin, LOW);
//deactivate segment E
digitalWrite(F_pin, HIGH);
//activate segment F
digitalWrite(G_pin, HIGH);
//activate segment G
break;

case 6:
/* display 6

     |
      -
     |  |
      -
  */
digitalWrite(A_pin, LOW);
//deactivate segment A
digitalWrite(B_pin, LOW);
//deactivate segment B
digitalWrite(C_pin, HIGH);
//activate segment C
digitalWrite(D_pin, HIGH);
//activate segment D
digitalWrite(E_pin, HIGH);
//activate segment E
digitalWrite(F_pin, HIGH);
//activate segment F
digitalWrite(G_pin, HIGH);
//activate segment G
break;

case 7:
/* display 7

      _
       |
       |

  */
digitalWrite(A_pin, HIGH);
//activate segment A
digitalWrite(B_pin, HIGH);
//activate segment B
digitalWrite(C_pin, HIGH);
//activate segment C
digitalWrite(D_pin, LOW);
//deactivate segment D
digitalWrite(E_pin, LOW);
//deactivate segment E
digitalWrite(F_pin, LOW);
//deactivate segment F
digitalWrite(G_pin, LOW);
//deactivate segment G
```

```
  break;

  case 8:
  /* display 8
         -
       |   |
         -
       |   |
         -

    */
  digitalWrite(A_pin, HIGH);
  //activate segment A
  digitalWrite(B_pin, HIGH);
  //activate segment B
  digitalWrite(C_pin, HIGH);
  //activate segment C
  digitalWrite(D_pin, HIGH);
  //activate segment D
  digitalWrite(E_pin, HIGH);
  //activate segment E
  digitalWrite(F_pin, HIGH);
  //activate segment F
  digitalWrite(G_pin, HIGH);
  //activate segment G
  break;

  case 9:
  /* display 9
         -
       |   |
         -
           |

    */
  digitalWrite(A_pin, HIGH);
  //activate segment A
  digitalWrite(B_pin, HIGH);
  //activate segment B
  digitalWrite(C_pin, HIGH);
  //activate segment C
  digitalWrite(D_pin, LOW);
  //deactivate segment D
  digitalWrite(E_pin, LOW);
  //deactivate segment E
  digitalWrite(F_pin, HIGH);
  //activate segment F
  digitalWrite(G_pin, HIGH);
  //activate segment G
  break;
  }
}
```

Step 3 (5 minutes)	Run the simulation and check the correct operation of the circuit

Step 4 (5 minutes)	Suggested modifications and discussion: • Can the same code work with a common anode seven segment display? • If the numbers change every 10ms, what will be displayed?

Chapter 3: **Recapitulation**

The circuits were designed and simulated with Tinkercad. Basic Arduino Uno programming functions were used, such as:

- pinMode()
- delay()
- analogWrite()
- digitalWrite()

Through the activities, Arduino Uno pins were used as output to lead:

- buzzer
- LED
- RGB LED
- Seven segment display

References

Breadboard \ Wiring. Retrieved from http://wiring.org.co/learning/tutorials/breadboard/

Brown, R. (2020). *Active vs. Passive buzzer: the differences*. Retrieved from https://nerdytechy.com/active-vs-passive-buzzer/

Learn C - Free Interactive C Tutorial. Retrieved from https://www.learn-c.org/

Learn how to use Tinkercad | Tinkercad. Retrieved from https://www.tinkercad.com/learn/circuits