ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

---

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

Practice leaflet: Module_1-2 pins as inputs

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 Where other published and unpublished source materials have been used, these have been acknowledged.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 inputs.

Chapter 1: **Overview**

Table 1. Overview

Title / short summary	**Pins as inputs: switches and push-buttons** **In this lesson we will use Arduino Uno pins as inputs.**
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 inputs • Design and implement simple circuits with switches and push-buttons
Keywords	Input pins, switch, push-button
Duration	The duration of the module_1-2 is 3 hours • Module_1-2 slides - 30 minutes • 1st activity: switches - 50 minutes • 2nd activity: push-buttons - 55 minutes • 3rd activity: switch and push-button - 45 minutes

Involved	**The students**: • Take part in activities • Complete circuit • Answer questionnaires **The teachers**: • Show the presentation of the module • Answer questions • Point out the tips • Encourage participation and discussion
Assignment	The module_1-2 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 and Module_1-2

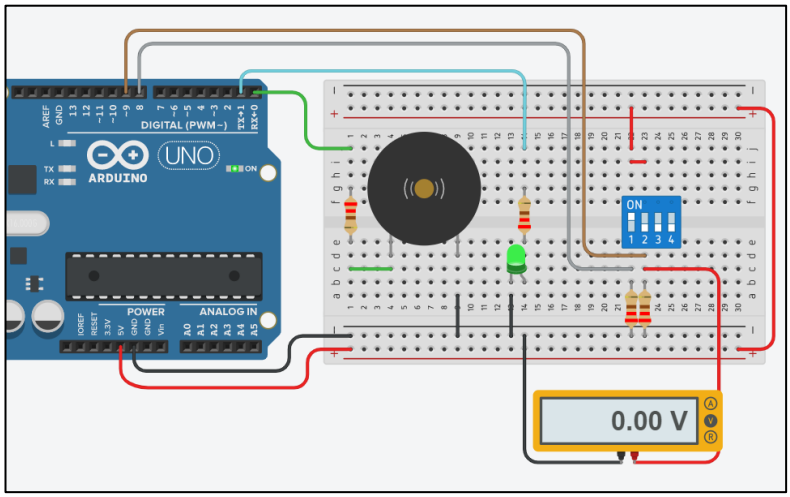
Educational content	Accompanying material: • Module_1-2 slides • Module_1-2 Evaluation leaflet • Module_1-2 Open Projects
Tips	***Tip 1***. When the switches are open the Arduino Uno reads “0”, while when they are closed the Arduino Uno reads “1” - pay attention to the wiring/topology ***Tip 2***. The voltmeter is always connected in parallel with the circuit ***Tip 3***. The RGB LED in Tinkercad is a common cathode ***Tip 4***. For the button: if it is not pressed, terminal_1a is short-circuited with terminal_1b and terminal_2a with terminal_2b. When it is pressed, all 4 ends are short-circuited

Chapter 2: **Activities**

2.1 Activity 1. Switches

This activity uses switches that provide input signals on the Arduino Uno. The main objective is to understand the circuit wiring and the corresponding code.

Table 2. Activity 1

Activity 1a (20 minutes)	In this part the aim is for the Arduino Uno to read the states of 2 switches • A buzzer follows the state of the first input. That is, the buzzer is activated when "1" comes from the switch_1 • An LED follows the reverse state of the second input. That is, the LED is activated when "0" comes from the switch_2 • A voltmeter has been added to the circuit to check the voltage at the switch_2 **Step 1**. Draw the circuit in Tinkercad **Step 2**. Write the microcontroller code **Step 3**. Simulate the circuit and test it
Step 1 (8 minutes)	Draw the next circuit in Tinkercad.  *Figure 1. Switches, buzzer and LED*

Step 2
(10 minutes)
	Study the code and write it on the microcontroller:
	
	``
	/* Switches, buzzer and LED
	
	Circuit Connections:
	PIN_0 => Resistor 220Ω => Buzzer_Positive - Buzzer_Negative = > Gnd
	PIN_1 => Resistor 220Ω => LED_Anode - LED_Cathode = > Gnd
	PIN_8 => Pull down resistor (220Ω) => switch_1 (Vcc)
	PIN_9 => Pull down resistor (220Ω) => switch_2 (Vcc)
	*/
	
	#define Buzzer_pin 0 //give the name "Buzzer_pin" to PIN_0
	#define led_pin 1 //give the name "led_pin" to PIN_1
	#define Sw1_pin 8 //give the name "Sw1_pin" to PIN_8
	#define Sw2_pin 9 //give the name "Sw2_pin" to PIN_9
	
	//The setup() function initializes and sets the initial values
	//It will only run once after each powerup or reset
	void setup() {
	  //Configure PIN_0 and PIN_1 to behave as output
	  //Configure PIN_8 and PIN_9 to behave as input
	  pinMode(Buzzer_pin, OUTPUT);
	  pinMode(led_pin, OUTPUT);
	  pinMode(Sw1_pin, INPUT);
	  pinMode(Sw2_pin, INPUT);
	}
	
	//This function loops consecutively
	void loop() {
	  //The Buzzer follows the state of switch_1
	  digitalWrite(Buzzer_pin, digitalRead(Sw1_pin));
	  //The LED follows the invert state of switch_2
	  digitalWrite(led_pin, !digitalRead(Sw2_pin));
	}
	``
	
	Run the simulation and check the correct operation of the circuit
	
	***Tip**. According to the code, when the voltmeter shows 5V the LED stays off, while when it shows 0V, the LED is activated.*

Step 3 (2 minutes)	

Activity 1b
(30 minutes)

In this part the aim is for the Arduino Uno to read the states of 2 switches, and drive an RGB LED whose color depends on the combinations of input:

Possible states and RGB LED

Switch_1	Switch_4	RGB LED
0	0	OFF
0	1	Red
1	0	Green
1	1	Blue

Step 1. Draw the circuit in Tinkercad

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.

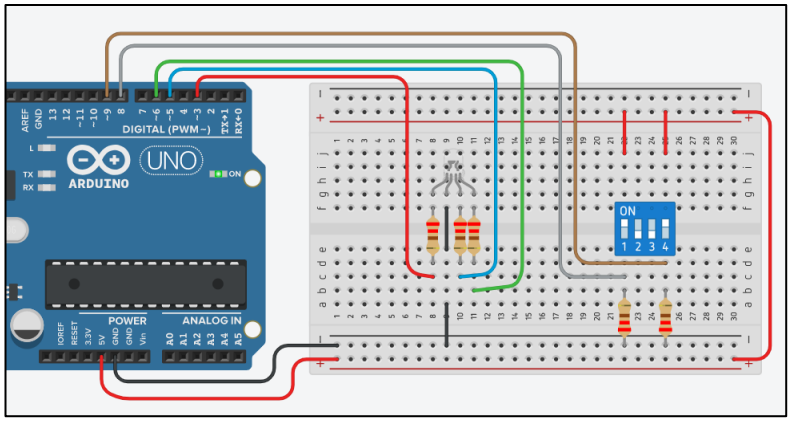


Figure 2. Switches and RGB LED

Step 2 (15 minutes)	Study the code and write it on the microcontroller:

```
/* Switches and RGB LED

Circuit Connections:
PIN_3 => Resistor 220Ω =>  Red pin of RGB LED
PIN_5 => Resistor 220Ω =>  Blue pin of RGB LED
PIN_6 => Resistor 220Ω =>  Green pin of RGB LED
PIN_8  =>  Pull  down  resistor  (220Ω)  =>  switch_1
(Vcc)
PIN_9  =>  Pull  down  resistor  (220Ω)  =>  switch_4
(Vcc)
*/

#define R_pin 3         //give the name "R_pin" to
PIN_3
#define G_pin 6         //give the name "G_pin" to
PIN_6
#define B_pin 5           //give the name "B_pin"
to PIN_5
#define Sw1_pin 8     //give the name "Sw1_pin" to
PIN_8
#define Sw4_pin 9     //give the name "Sw4_pin" to
PIN_9

//The  setup()  function  initializes  and  sets  the
initial values
//It will only run once after each powerup or reset
void setup()
{
  //Configure PIN_3, PIN_5 and PIN_6 to behave as
output
  //Configure PIN_8 and PIN_9 to behave as input
  pinMode(R_pin, OUTPUT);
  pinMode(G_pin, OUTPUT);
  pinMode(B_pin, OUTPUT);
  pinMode(Sw1_pin, INPUT);
  pinMode(Sw4_pin, INPUT);
}

//This function loops consecutively
void loop() {
  if(digitalRead(Sw1_pin)==0                    &&
digitalRead(Sw4_pin)==0){
      //RGB LED is OFF
      analogWrite(R_pin, 0);     //Write 0% PWM to
pin 3
      analogWrite(G_pin, 0);     //Write 0% PWM to
pin 6
      analogWrite(B_pin, 0);     //Write 0% PWM to
pin 5
      delay(1000);                          // Wait
for 1 second
  }
  else        if(digitalRead(Sw1_pin)==0        &&
digitalRead(Sw4_pin)==1){
      //red color for RGB = > R=255, G=0, B=0
      analogWrite(R_pin, 255); //Write 100% PWM to
pin 3
```

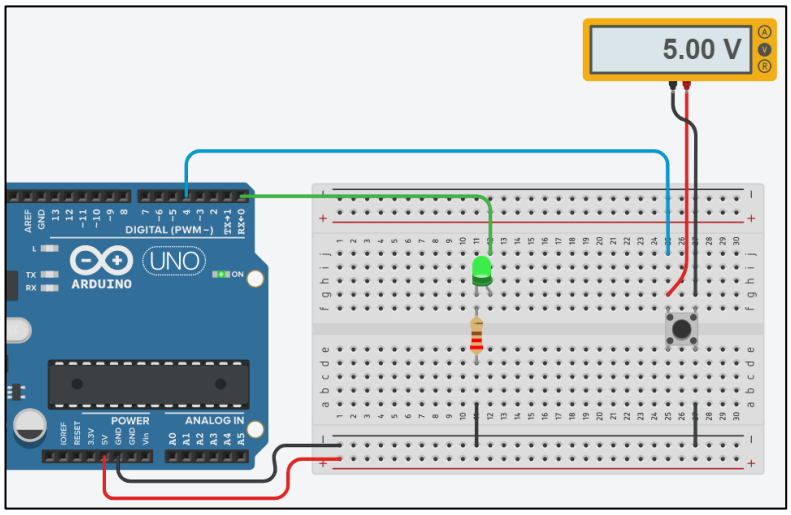
	`analogWrite(G_pin, 0);  //Write 0% PWM to pin 6`
	`analogWrite(B_pin, 0);  //Write 0% PWM to pin 5`
	`delay(1000);  // Wait for 1 second`
	`}`
	`else  if(digitalRead(Sw1_pin)==1  && digitalRead(Sw4_pin)==0){`
	`//green color for RGB = > R=0, G=255, B=0`
	`analogWrite(R_pin, 0);  //Write 0% PWM to pin 3`
	`analogWrite(G_pin, 255); //Write 100% PWM to pin 6`
	`analogWrite(B_pin, 0);  //Write 0% PWM to pin 5`
	`delay(1000);  // Wait for 1 second`
	`}`
	`else{  //if(digitalRead(Sw1_pin)==1  && digitalRead(Sw4_pin)==1)`
	`//blue color = > RGB=0,0,255`
	`analogWrite(R_pin, 0);  //Write 0% PWM to pin 3`
	`analogWrite(G_pin, 0);  //Write 0% PWM to pin 6`
	`analogWrite(B_pin, 255); //Write 100% PWM to pin 5`
	`delay(1000);  // Wait for 1 second`
	`}`
	`}`

Step 3 (2 minutes)	Run the simulation and check the correct operation of the circuit
Step 4 (5 minutes)	Suggested modifications and discussion: • What should be changed in the **inputs** of the Arduino Uno so that the RGB LED instead of red, green, and blue illuminates pink, yellow and white? • What should be **added** to switches 2 and 3 so that their states can be read by the Arduino Uno?

2.2 Activity 2. Push-buttons

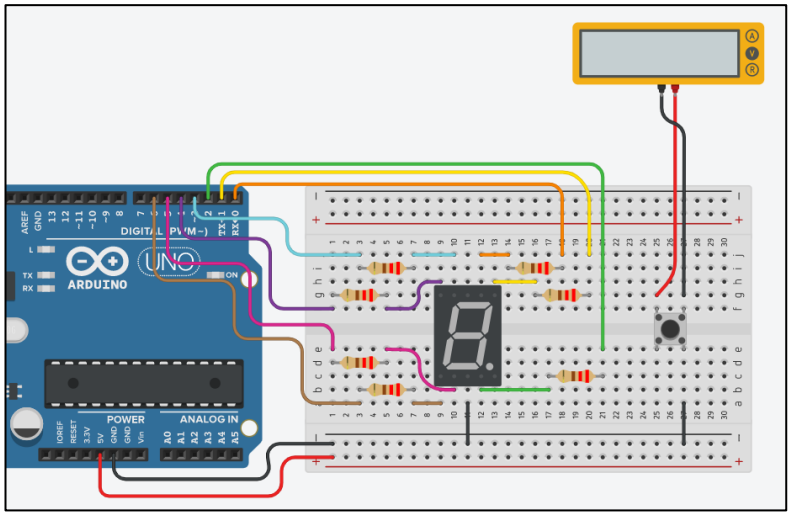
This activity uses push-button that provide input signals on the Arduino Uno. The main objective is to understand the circuit wiring and the corresponding code.

Table 3. Activity 2

Activity 2a (15 minutes)	In this part the aim is for the Arduino Uno to read the states of a push-button. • An LED follows the states of the push-button • The push-button is connected to PIN_4. The built-in **pull-up** resistor is activated with an appropriate setting in pinMode(), so no external resistor needs to be used • A voltmeter has been added to the circuit to check the voltage at the PIN_4 ***Tip***. *With this connection when the push-button is pressed the Arduino Uno reads "0" to the input and the LED is turned off* **Step 1**. Draw the circuit in Tinkercad **Step 2**. Write the microcontroller code **Step 3**. Simulate the circuit and test it
Step 1 (6 minutes)	Draw the next circuit in Tinkercad.  *Figure 3. Push-button and LED*

Step 2
(7 minutes)
	Study the code and write it on the microcontroller:
	
	``
	/* Push-button and LED
	
	Circuit Connections:
	PIN_0 => LED_Anode - LED_Cathode = > Resistor 220Ω
	=> Gnd
	PIN_4 => Pull-up resistor (built in) => push-button
	(Gnd)
	*/
	
	#define led_pin 0 //give the name “led_pin” to
	PIN_0
	#define pb_pin 4 //give the name “pb_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()
	{
	  //Configure PIN_0 to behave as outputs
	  pinMode(led_pin, OUTPUT);
	  //Configure PIN_4 to behave as input with
	activated pull-up resistor
	  pinMode(pb_pin, INPUT_PULLUP);
	}
	
	//This function loops consecutively
	void loop() {
	  //The LED follows the state of push-button
	  digitalWrite(led_pin, digitalRead(pb_pin));
	}
	``
	
	Run the simulation and check the correct operation of the circuit

Step 3 (2 minutes)	

Activity 2b (40 minutes)	In this part the aim is for the Arduino Uno to read the states of a push-button. When the push-button is **pressed and released**, the Arduino Uno counts from 0 to 9 on a common cathode seven segment display, with numbers changing every 500ms. • The push-button is connected to PIN_7. The built-in **pull-up** resistor is activated with an appropriate setting in pinMode(), so no external resistor needs to be used • A voltmeter has been added to the circuit to check the voltage at the PIN_7 **Step 1**. Draw the circuit in Tinkercad **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 4. Push-button and seven segment display*

Step 2 (18 minutes)	Study the code and write it on the microcontroller:

```
/* Push button and 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
PIN_7 => Pull-up resistor (built in) => push-button
(Gnd)
*/

#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
#define pb_pin 7          //give the name "pb_pin"
to PIN_7

boolean pressAndReleased=false;       //flag for
push-button

//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
  //Configure PIN_7 to behave as input with
activated pull-up resistor
  pinMode(pb_pin, INPUT_PULLUP);
}
```

```
//This function loops consecutively
void loop() {
  if    (digitalRead(pb_pin)==0){    //push-button
pressed
  delay(25);  //debounce
      while(digitalRead(pb_pin)==0){;}
      //push-button released
      delay(25);    //debounce
      //set the flag to true
      pressAndReleased=true;
  }
  //check  the  flag  for  push-button  press  and
release
  if (pressAndReleased == true){
      //call  the  function  "sevenSegment"  and
display the numbers from 0 to 9
      for (int i=0; i<10; i++){
           sevenSegment(i);
           delay(500);     //wait for 0.5 second
      }
      //set the flag to false
      pressAndReleased =false;
      //deactivate every segment
      digitalWrite(A_pin, LOW);
      digitalWrite(B_pin, LOW);
      digitalWrite(C_pin, LOW);
      digitalWrite(D_pin, LOW);
      digitalWrite(E_pin, LOW);
      digitalWrite(F_pin, LOW);
      digitalWrite(G_pin, LOW);
  }
}

//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 (2 minutes)	Run the simulation and check the correct operation of the circuit
Step 4 (5 minutes)	Suggested modifications and discussion: • What should be changed for the input pin if the built-in pull-up resistor was not activated? • What should be changed so that when the push-button is pressed the Arduino Uno reads “1” (5V)?

2.3 Activity 3. Seven segment display

This activity uses push-button and switches that provide input signals on the Arduino Uno. The main objective is to understand the circuit wiring and the corresponding code.

Table 4. Activity 3

Activity 3
(45 minutes)

When the push-button is **pressed and released**, the Arduino Uno counts from 0 to 9 on a common cathode seven segment display. The numbers change at a rate defined by switches states, Table 2.

- The push-button is connected to PIN_7. The built-in **pull-up** resistor is activated with an appropriate setting in pinMode(), so no external resistor needs to be used
- A voltmeter has been added to the circuit to check the voltage at the PIN_7

States and time delay setting

Switch_1	Switch_2	Switch_3	delay_ms
0	0	0	200
0	0	1	2000
1	1	0	400
1	1	1	4000
All other combinations			500

Step 1. Draw the circuit in Tinkercad

Step 2. Write the microcontroller code

Step 3. Simulate the circuit and test it

Step 1
(20 minutes)

Draw the next circuit in Tinkercad.

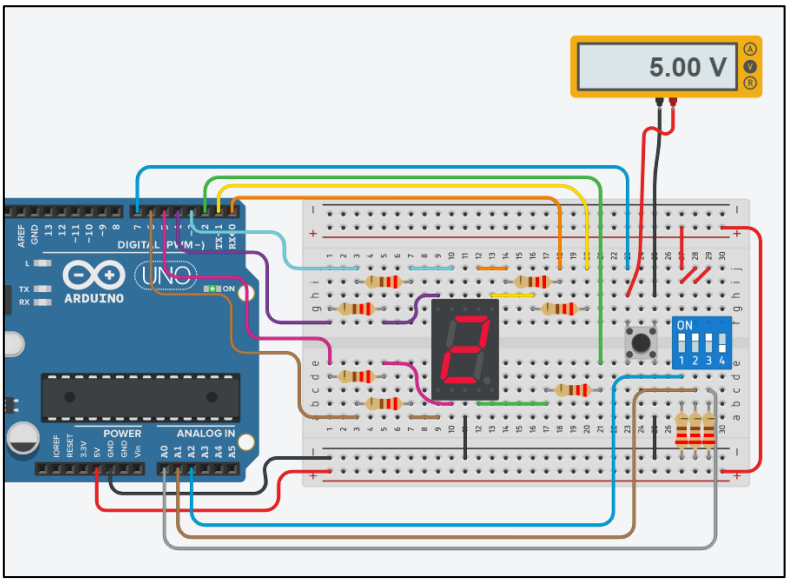


Figure 5. Switches, push-button and seven segment display

Step 2 (20 minutes)	Study the code and write it on the microcontroller:

```
/* Switches, push-button and 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
PIN_7 => Pull-up resistor (built in) => push-button
(Gnd)
PIN_A2 => Switch_1 (Vcc)
PIN_A1 => Switch_2 (Vcc)
PIN_A0 => Switch_3 (Vcc)
*/

#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
#define pb_pin 7          //give the name "pb_pin"
to PIN_7
#define sw1_pin A2  //give the name "sw1_pin_ to
PIN_A2
#define sw2_pin A1  //give the name "sw1_pin_ to
PIN_A1
#define sw3_pin A0  //give the name "sw1_pin_ to
PIN_A0

boolean pressAndReleased=false;       //flag for
push-button
int speed;                        //variable for
delay time

//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
  //Configure  PIN_7  to  behave  as  input  with
activated pull-up resistor
  pinMode(pb_pin, INPUT_PULLUP);
  //Configure PIN_A0, PIN_A1 and PIN_A2 to behave
as inputs
  pinMode(sw1_pin, INPUT);
  pinMode(sw2_pin, INPUT);
  pinMode(sw3_pin, INPUT);
}

//This function loops consecutively
void loop() {
  //check switches for speed settings
  if          (digitalRead(sw1_pin)==0          &&
digitalRead(sw2_pin)==0){
    speed=200;
    if(digitalRead(sw3_pin)==1){
      speed=2000;
    }
  }
  else        if(digitalRead(sw1_pin)==1        &&
digitalRead(sw2_pin)==1){
    speed=400;
    if(digitalRead(sw3_pin)==1){
      speed=4000;
    }
  }
  else{
    speed=500;
  }
  if    (digitalRead(pb_pin)==0){    //push-button
pressed
  delay(25);  //debounce
      while(digitalRead(pb_pin)==0){;}
      //push-button released
      delay(25);    //debounce
      //set the flag to true
      pressAndReleased=true;
  }
  //check  the  flag  for  push-button  press  and
release
  if (pressAndReleased == true){
      //call  the  function  "sevenSegment"  and
display the numbers from 0 to 9
      for (int i=0; i<10; i++){
           sevenSegment(i);
           delay(speed);       //wait for "speed"
milliseconds
      }
      //set the flag to false
      pressAndReleased =false;
      //deactivate every segment
      digitalWrite(A_pin, LOW);
      digitalWrite(B_pin, LOW);
```

```
        digitalWrite(C_pin, LOW);
        digitalWrite(D_pin, LOW);
        digitalWrite(E_pin, LOW);
        digitalWrite(F_pin, LOW);
        digitalWrite(G_pin, LOW);
  }
}

//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

Chapter 3: **Recapitulation**

The circuits were designed and simulated with Tinkercad.

Basic Arduino Uno programming functions were used, such as:

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

Through the activities, Arduino Uno pins were used as inputs to read states from:

- switches
- push-button

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