ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

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

Practice leaflet: Module_1-3 communication and ADC

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 Where other published and unpublished source materials have been used, these have been acknowledged.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 Serial communication and Analog to Digital Converter.

Chapter 1: **Overview**

Table 1. Overview

Title / short summary	**Communication and ADC:** **Serial communication and Analog to Digital Converter**
Expected learning outcomes	Students completing the course will be able to: • Recognize basic Arduino Uno functions and programming structures • Use the Arduino Uno's two-way serial communication with the monitor • Handle analog signals with the Arduino Uno • Design and implement simple circuits with serial communication and analog
Keywords	Serial communication, ADC
Duration	The duration of the module_1-3 is 3 hours • Module_1-3 slides - 30 minutes • 1st activity: Serial and LEDs - 50 minutes • 2nd activity: Read Serial data - 50 minutes • 3rd activity: ADC- 50 minutes

Involved	**The students**: • Take part in activities • Complete code • Answer questionnaires **The teachers**: • Show the presentation of the module • Answer questions • Point out the tips • Encourage participation and discussion
Assignment	The module_1-3 includes: • 3 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, Module_1-2 and Module_1-3

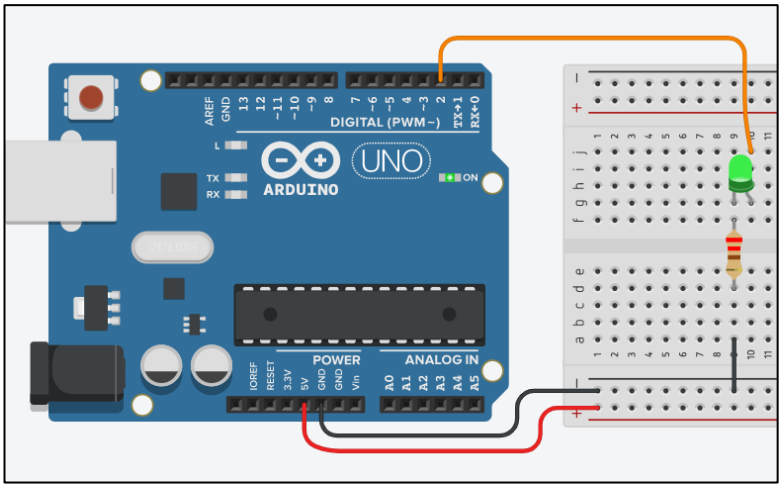
Educational content	Accompanying material: • Module_1-3 slides • Module_1-3 Evaluation leaflet • Module_1-3 Open Projects
Tips	***Tip***. *ADC resolution is 10 bits (number range: 0 ~ 1023)*

Chapter 2: **Activities**

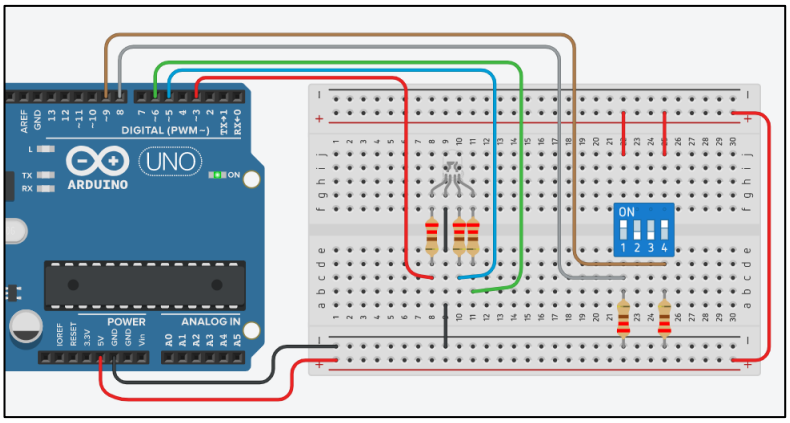
2.1 Activity 1. Serial and LEDs

This activity uses serial communication between the Arduino Uno and the monitor.

Table 2. Activity 1

Activity 1a (15 minutes)	In this part the aim is for the Arduino Uno to flash an LED every 5 seconds. Each time the LED changes status, the Arduino Uno sends a corresponding message to the serial. • The data sent by the Arduino Uno via the serial port is displayed on the Tinkercad monitor **Step 1**. Draw the circuit in Tinkercad **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. LED and serial communication*

Step 2 (8 minutes)	Study the code and write it on the microcontroller: `/* Blinking a LED and print to Serial` `Circuit Connections:` `PIN_2 => LED_Anode - LED_Cathode = > Resistor 220Ω => Gnd` `PIN_0 => Serial_RX` `PIN_1 => Serial_TX` `*/` `//The setup() function initializes and sets the initial values` `//It will only run once after each power up or reset` `void setup()` `{` `  //Configure the PIN_2 to behave as output` `  pinMode(2, OUTPUT);` `  //opens serial port, sets data rate to 9600 bps` `  Serial.begin(9600);` `}` `//loops consecutively` `void loop()` `{` `  digitalWrite(2, HIGH); //Write a HIGH value (5V) to a digital pin` `  Serial.println("LED is ON");//sent data` `  delay(5000); // Pauses the program for 5000 milliseconds` `  digitalWrite(2, LOW); //Write a LOW value (0V) to a digital pin` `  Serial.println("LED is OFF");//sent data` `  delay(5000); // Wait for 5000 milliseconds` `}`
Step 3 (2 minutes)	Run the simulation and check the correct operation of the circuit ***Tip**. Open the Tinkercad monitor to see the data.*

Activity 1b (35 minutes)	In this part the aim is for the Arduino Uno to: • read the states of 2 switches • drive an RGB LED whose color depends on the combinations of input • inform via serial communication about the color changes of the RGB LED 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, RGB LED and serial*

Step 2
(20 minutes)

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

```
/* Switches, RGB LED and Serial

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)

PIN_0 => Serial_RX
PIN_1 => Serial_TX
*/

#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  power  up  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);
  //opens serial port, sets data rate to 9600 bps
=>
}

//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
      Serial.println("LED is OFF");  //sent data
```

```
      //as long as the state of the inputs does
not change
      while(digitalRead(Sw1_pin)==0            &&
digitalRead(Sw4_pin)==0){;}
  }
  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
      Serial.println("LED is RED");  //sent data
      //as long as the state of the inputs does
not change
      while(digitalRead(Sw1_pin)==0            &&
digitalRead(Sw4_pin)==1){;}
  }
  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
      Serial.println("LED is GREEN");  //sent data
      //as long as the state of the inputs does
not change
=>

  }
  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
      Serial.println("LED is BLUE");  //sent data
      //as long as the state of the inputs does
not change
      while(digitalRead(Sw1_pin)==1            &&
digitalRead(Sw4_pin)==1){;}
  }
}
```

Step 3 (2 minutes)	Run the simulation and check the correct operation of the circuit

Step 4 (5 minutes)	Suggested modifications and discussion: • Could components be connected to PIN_0, PIN_1?

2.2 Activity 2. Push-buttons

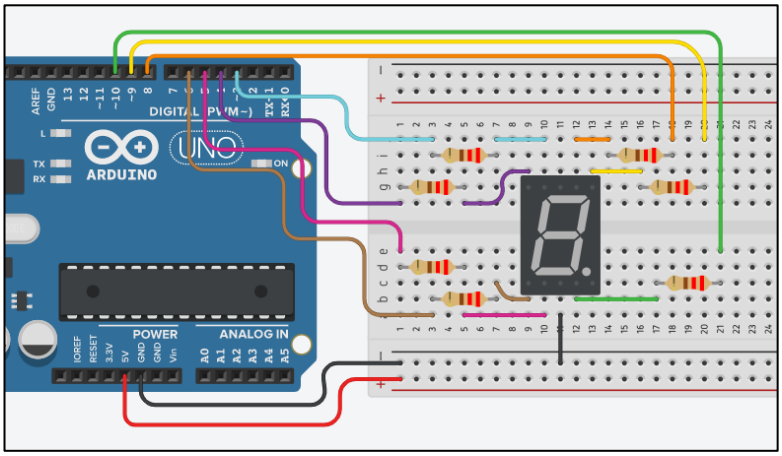
In this activity the Arduino Uno reads data from the serial communication and performs actions according to the commands it has read. More specifically, the Arduino Uno works as an up / down-counter that counts from 0 to 9 every 500ms. The numbers are shown in a 7 segment display. Setting the counter to count up or down is done by serial communication.

Table 3. Activity 2

<table>
<tr>
<td>Activity 2
(50 minutes)</td>
<td>By default, the Arduino Uno works as an up-counter. It can accept 2 commands from serial communication.

Commands from the serial communication
<table>
<tr><th>Command</th><th>Selection</th></tr>
<tr><td>“up”</td><td>Up-counter</td></tr>
<tr><td>“down”</td><td>Down-counter</td></tr>
<tr><td colspan="2">Anything else is considered a wrong command</td></tr>
</table>

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</td>
</tr>
</table>

Step 1 (15 minutes)	Draw the next circuit in Tinkercad.  *Figure 3. Serial and seven segment display*

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

```
/* Up/down-counter and seven segment display

Circuit Connections:
Seven segment common Cathode = > Gnd
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_8   =>  Resistor 220Ω => Segment a
PIN_9   =>  Resistor 220Ω => Segment b
PIN_10 =>  Resistor 220Ω => Segment c

PIN_0 => Serial_RX
PIN_1 => Serial_TX
*/

#define A_pin 8     //give  the  name  "A_pin"  to
PIN_8
#define B_pin 9     //give  the  name  "B_pin"  to
PIN_9
#define C_pin 10    //give  the  name  "C_pin"  to
PIN_10
#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

boolean  count=true;           //true=up-counter,
false=down-counter
String input; //variable to save data from serial
int i=0; // variable to hold the number for the
seven segment display

//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_8
to behave as output
  pinMode(B_pin, OUTPUT);   //Configure the PIN_9
to behave as output
  pinMode(C_pin, OUTPUT);  //Configure the PIN_10
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
  //opens serial port, sets data rate to 9600 bps
  Serial.begin(9600);
```

```
}


//This function loops consecutively
void loop() {
  //call the function "sevenSegment" and display
the number "i"
  sevenSegment(i);
  delay(500);  //wait for 0.5s

  //check for serial data
  if (Serial.available() > 0){
    //read and save data
    input = Serial.readString();
    //check data's value
    if(input == "up"){
      count=true; //up-counter
    }
    else if(input=="down"){
      count=false;  //down-counter
    }
   else{
     Serial.println("Wrong command");
    }
  }


//increase (or decrease) the number and check for
overflow
  if (count == true){
    i++;
    if(i>9){
      i=0;
    }
  }
  else{
    i--;
    if(i<0){
      i=9;
    }
  }
}

//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
```

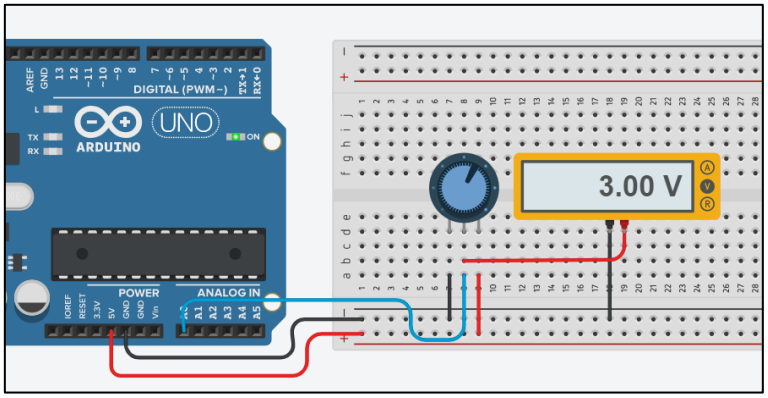
```
      digitalWrite(G_pin, HIGH);
      //activate segment G
      break;

      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 (10 minutes)	Suggested modifications and discussion: • Add commands via serial communication: start / pause counter

2.3 Activity 3. Analog to Digital Converter

This activity uses the Arduino Uno's built-in analog-to-digital converter.

Table 4. Activity 3

Activity 3a (20 minutes)	In this part the aim is for the Arduino Uno to: • read the analog voltage of a potentiometer • send it to serial communication • A voltmeter has been added to the circuit to check the voltage of the potentiometer **Step 1**. Draw the circuit in Tinkercad **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 4. ADC and Serial communication*

Step 2 (10 minutes)	Study the code and write it on the microcontroller: `/* ADC and serial` `Circuit connections:` `Potensiometer_Terminal_1 => Gnd` `Potensiometer_Wiper => A0` `Potensiometer_Terminal_2 => Vcc` `PIN_0 => Serial_RX` `PIN_1 => Serial_TX` `*/` `#define pot_pin A0  //give the name "pot_pin" to PIN_A0` `//variable to save data from ADC` `int adc_value; //number range 0~1023` `float voltage;  //variable to calculate the analog voltage` `//The setup() function initializes and sets the initial values` `//It will only run once after each power up or reset` `void setup() {` `  Serial.begin(9600);` `}` `//This function loops consecutively` `void loop() {` `  //read analog voltage and convert to number` `  adc_value = analogRead(pot_pin);` `  //calculate the analog voltage from adc_number` `  voltage = float(adc_value)/1023*5;` `  //print to serial the adc_number and the analog voltage` `  Serial.print("ADC number: ");` `  Serial.println(adc_value);` `  Serial.print("Voltage = ");` `  Serial.print(voltage);` `  Serial.println("V");` `  //wait for 5 seconds` `  delay(5000);` `}`
Step 3 (5 minutes)	Run the simulation and check the correct operation of the circuit

Activity 3b
(30 minutes)

In this part the aim is for the Arduino Uno to:

- read the analog voltage of a potentiometer
- divide it into 5 equal parts
- turn on / off 5 LEDs according to the table below
- A voltmeter has been added to the circuit to check the voltage of the potentiometer

Analog voltage and LEDs

Vout_pot (V)	Activated LEDs
0 ~ 1	1
1 ~ 2	2
2 ~ 3	3
3 ~ 4	4
4 ~ 5	5

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
(10 minutes)

Draw the next circuit in Tinkercad.

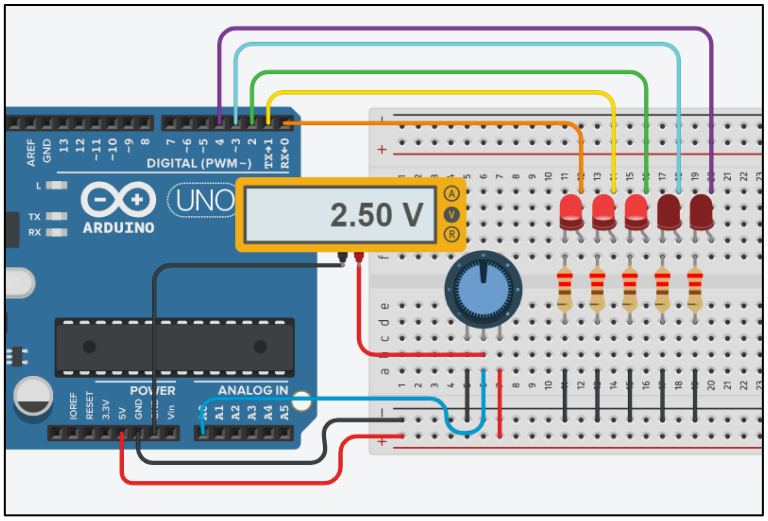


Figure 5. ADC and LEDs

Step 2 (12 minutes)	Study the code and write it on the microcontroller:

```
/* ADC and LEDs

Circuit connections:
Potensiometer_Terminal_1 => Gnd
Potensiometer_Wiper => A0
Potensiometer_Terminal_2 => Vcc

PIN_0 => Resistor 220Ω => LED1_Anode – LED1_Cathode
= > Gnd
PIN_1 => Resistor 220Ω => LED2_Anode – LED2_Cathode
= > Gnd
PIN_2 => Resistor 220Ω => LED3_Anode – LED3_Cathode
= > Gnd
PIN_3 => Resistor 220Ω => LED4_Anode – LED4_Cathode
= > Gnd
PIN_4 => Resistor 220Ω => LED5_Anode – LED5_Cathode
= > Gnd
*/

#define LED1_pin 0  //give the name "LED1_pin" to
PIN_0
#define LED2_pin 1  //give the name "LED2_pin" to
PIN_1
#define LED3_pin 2  //give the name "LED3_pin" to
PIN_2
#define LED4_pin 3  //give the name "LED4_pin" to
PIN_3
#define LED5_pin 4  //give the name "LED5_pin" to
PIN_4
#define pot_pin A0  //give the name "pot_pin" to
PIN_A0
//variable to save data from ADC
int adc_value; //number range 0~1023
//variable to calculate the analog voltage

//The setup() function initializes and sets the
initial values
//It will only run once after each power up or
reset
void setup() {
  //Configure PIN_0, PIN_1, PIN_2, PIN_3 and  PIN_4
to behave as output
  pinMode(LED1_pin, OUTPUT);
  pinMode(LED2_pin, OUTPUT);
  pinMode(LED3_pin, OUTPUT);
  pinMode(LED4_pin, OUTPUT);
  pinMode(LED5_pin, OUTPUT);
}

//This function loops consecutively
void loop() {
  //read analog voltage and convert to number
  adc_value = analogRead(pot_pin);
  //check if voltage < 1V
  if(adc_value<204){
      //activate LED1
   digitalWrite(LED1_pin, HIGH);
```

	`` digitalWrite(LED2_pin, LOW); digitalWrite(LED3_pin, LOW); digitalWrite(LED4_pin, LOW); digitalWrite(LED5_pin, LOW); } //check if voltage < 2V else if(adc_value<408){ //activate LED1 and LED2 digitalWrite(LED1_pin, HIGH); digitalWrite(LED2_pin, HIGH); digitalWrite(LED3_pin, LOW); digitalWrite(LED4_pin, LOW); digitalWrite(LED5_pin, LOW); } //check if voltage < 3V else if (adc_value<612){ //activate LED1, LED2 and LED3 digitalWrite(LED1_pin, HIGH); digitalWrite(LED2_pin, HIGH); digitalWrite(LED3_pin, HIGH); digitalWrite(LED4_pin, LOW); digitalWrite(LED5_pin, LOW); } //check if voltage < 4V else if(adc_value<816){ //activate LED1, LED2, LED3 and LED4 digitalWrite(LED1_pin, HIGH); digitalWrite(LED2_pin, HIGH); digitalWrite(LED3_pin, HIGH); digitalWrite(LED4_pin, HIGH); digitalWrite(LED5_pin, LOW); } // if voltage < 5V else{//if(adc_value<1023){ //activate all LEDs digitalWrite(LED1_pin, HIGH); digitalWrite(LED2_pin, HIGH); digitalWrite(LED3_pin, HIGH); digitalWrite(LED4_pin, HIGH); digitalWrite(LED5_pin, HIGH); } delay(250); //wait for 250ms } ```
Step 3 (3 minutes)	Run the simulation and check the correct operation of the circuit
Step 4 (5 minutes)	Suggested modifications and discussion: • In this activity we divided the analog values of the potentiometer into 5 equal parts. What is the maximum number of parts we can divide the potentiometer values?

Chapter 3: **Recapitulation**

The circuits were designed and simulated with Tinkercad.

Basic Arduino Uno programming functions were used, such as:

- delay()
- analogWrite()
- digitalWrite()
- analogRead()
- Serial.begin()
- Serial.abailable()

Through the activities were utilized

- Arduino Uno pins as analog inputs
- Two-way serial communication

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