ELEN30013 Workshop:
Arduino Supplemental (not assessed)

1. **Serial, Character & String (not assessed)**
2. **Arrays with LEDs (not assessed)**

1. Serial, Character & String

The following is a list of the key documentation needed for this section:

https://www.arduino.cc/reference/en/language/functions/communication/serial/

https://www.arduino.cc/reference/en/language/variables/data-types/char/

https://www.arduino.cc/reference/en/language/variables/data-types/string/

Serial Functions:

if(Serial)	available()	availableForWrite()
begin()	end()	find()
findUntil()	flush()	parseFloat()
parseInt()	peek()	print()
println()	read()	readBytes()
readBytesUntil()	readString()	readStringUntil()
setTimeout()	write()	serialEvent()

Task 1:

In this task we will go over the basics of initialising a serial object and using it for basic communication.

1. In the setup function initialise the serial with:

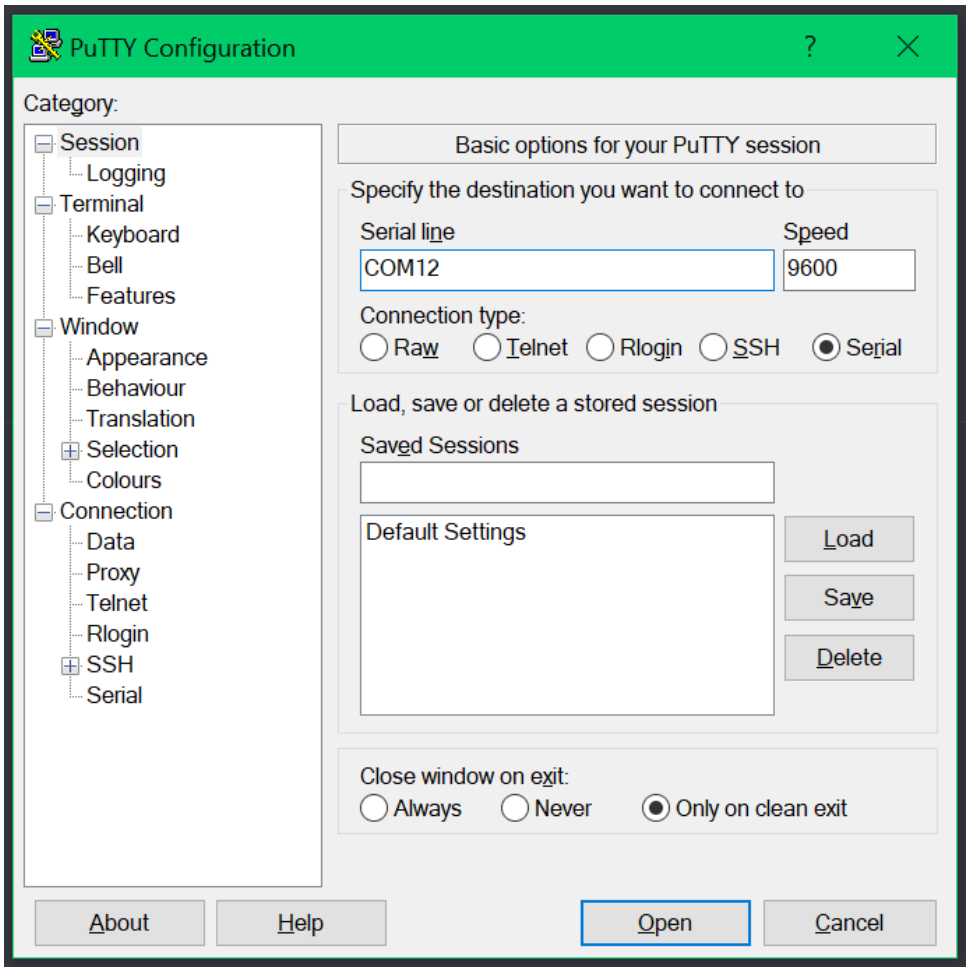
   ```
   Serial.begin(9600);
   ```

2. In the loop function make the Arduino send data to the PC with:

   ```
   Serial.println("ESI is awesome!!!");
   ```

3. Upload the code to your Arduino
4. View the serial print with the tools > serial monitor (shortcut: ctrl + shift + m)

5. Close the serial monitor and now repeat the previous step using putty (https://www.putty.org/)



You will need to set:

- Connection type: Serial
- Speed: 9600
- Serial line: “The Com Port your Arduino is connected to”

Task 2:

In this task we will investigate printing individual characters via Serial.

1. Create a global int variable called val with an initial value of 0.
2. Add the serial begin in the setup function with a buad rate of 9600.
3. In the loop create a for loop that will iterate a local variable i from 1 to 255. Then within each cycle of the for loop:
    a. Use the serial print line method to print val in a number of ways

```
Serial.print(val);
Serial.print(" ");
Serial.print(val, DEC);
Serial.print(" ");
Serial.print(val, HEX);
Serial.print(" ");
Serial.print(val, OCT);
Serial.print(" ");
Serial.println(val, BIN);
```

    b. Increment val by 1.
4. Upload the program to your Arduino and view the results in the serial monitor or with Putty. Can you explain what is happening, if not ask your demonstrator to clarify or consult your group.
5. Repeat the activity using char in place of int for the type of val and ensure you initialise it to a single character surrounded with single quotes.

Note: Single quotes are used for char and char array. Double quotes are used for String which is a abstraction of char array.

Task 3:

In this task we will move on to arrays and Strings.

1. As before add serial begin to the setup
2. Create the global variables

```
char mychar1 = 'A';
char mychar2 = 65;
char mychars[] = {'E','s','i','!'};
String myString = "ESI IS AWESOME!";
```

3. In the loop function print out each of the variables followed by a 0.5 second delay.
4. Upload the program to your Arduino and view the results in the serial monitor or with Putty. Can you explain what is happening, if not ask your demonstrator to clarify or consult your group.

Task 4:

In this task we will conduct a brief investigation of reading data sent to the Arduino via serial.

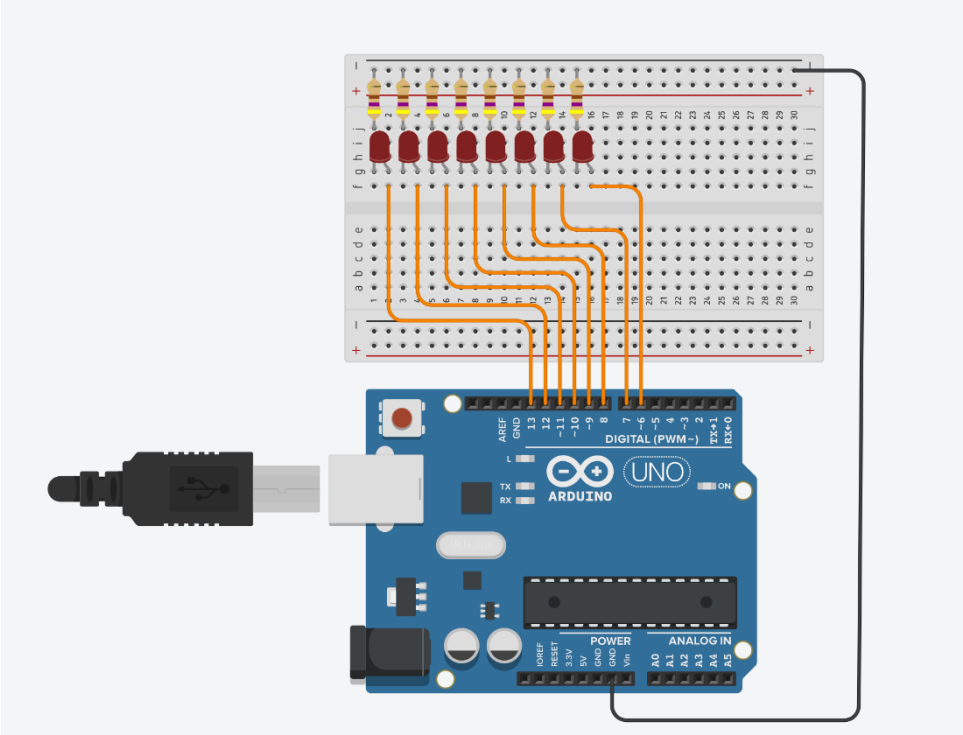
1. As before add serial begin to the setup function,
2. In the main loop function use an if function to check if Serial.available() is greater than zero
    a. Within this conditional use Serial.readStringUntil('\n') which will return a String reading up until the next new line character.
    b. Then have the a message that prints "ARD received:" followed by the message that it received.
3. Upload the program to your Arduino, send some text then via the serial monitor or with Putty. Can you explain what is happening, if not ask your demonstrator to clarify or consult your group.
4. Repeat the task using the following in place of Serial.readStringUntil('\n'):
    - Serial.read()
    - Serial. parseFloat()
    - Serial. parseInt()

    Note: you may need to refer to the Arduino Reference > Serial to determine the data types needed to store the methods result.

2. Arrays with LEDs:

https://www.arduino.cc/reference/en/language/variables/data-types/array/

Use the following diagram and code as a scaffold to solve a number of challenges involving LEDs and arrays.



```
const int mypins[] = {6,7,8,9,10,11,12,13};
const int numpins = 8;

void setup()
{
  Serial.begin(9600);
  for(int i=0; i<numpins; i++){
    pinMode(mypins[i],OUTPUT);
  }
}

void loop()
{
  delay(500);
}
```

T**ask 1:** Turn on all the LEDs using a for loop.

Task 2: Make the LEDs display in the pattern:

...0000

...0001

...0011

...0111

Stepping forward every 1 second. Where 0 is off and 1 is on.

Task 2: Make the LEDs display binary counting. Stepping forward every 0.2 second.

Task 3: Make the LEDs always have one and only 1 LED turned on. Every 1.5 seconds have the LED move one LED to the left and once it reaches the left most position have it jump back to the right most position before continuing.

Task 4: Make the LEDs always have one and only 1 LED turned on. Every 1.5 seconds have the LED move one LED to the left. Once it has reached the left most LED it will begin moving 1 LED to the right until it reaches the righthand side and returns to moving one LED to the left.

Task 5: Create an Arduino program that take in a number via serial and based on that numbers binary representation will turn on or off the LED corresponding to the nth bit being a 1 or a 0. You may which to investigate the function bitRead().

Task 6: Add 2 buttons to the circuit. Program the Arduino such that it every second the left button is pressed down it will move one LED to the left, if the left most LED is lit it will remain there. The same is true of the right button in the right direction.

Task 7: Create a 1-dimensional game using the LEDS. Conway's game of life or Falling sand may be fun or create your own!

Task 8: Using Tinkercad to keep the wiring easy create a circuit to investigate the 8x8 LED dot matrix and make use of the ideas covered in this workshop extending them to 2 dimensions!