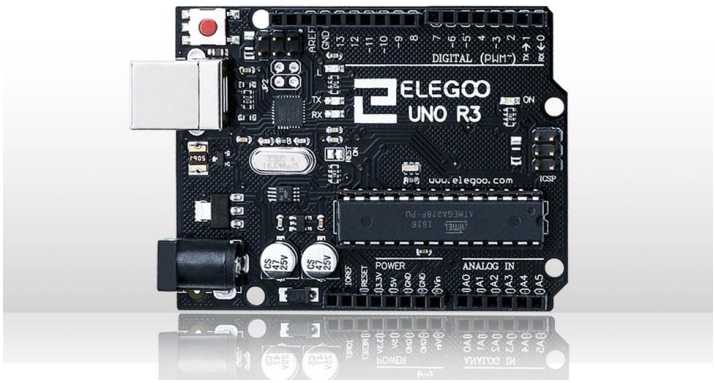
Serial Print on Set Interval

Arduino



Texas Tech University

Fall 2022

Description

This project demonstrates the ability to, on a user defined set interval, print serially using an Arduino an ASCII phrase or variable of choice. We have chosen to define the time interval at which a new line of text is printed to the serial port every 2 seconds, with our ASCII phrase of choice “Hello World”.

1. Introduction

This project will teach you how to develop a software application for the Arduino R3 board in C. The main purpose of this project is to introduce you to the basics of setting up a C project for an Arduino. With setting up comes the introduction installing software, choosing a baud rate and writing and understanding the code and use of the void loop function the Arduino IDE provides us. The basic premise of this project is to print “Hello World” every 2 seconds, as defined by our time interval.

2. Schematics

For our first project no schematics will be involved. As we progress further the projects will require additional items like a breadboard, LEDs and various sensors.

3. Hardware setup

The hardware setup for this project simply requires an Arduino R3 plugged into a computers USB port. The computer can run Windows, Linux or Mac OS as long as it supports the latest Arduino IDE software.

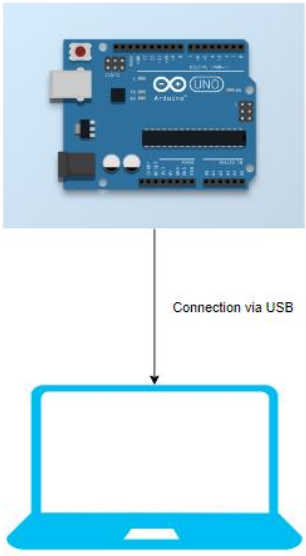


Figure 1: Basic Hardware Setup

4. Required software

The only required software for this project is the Arduino IDE which can be found here: https://www.arduino.cc/en/software

5. Software Setup

Software setup after installation is simple. Once you have opened up a clean new 'sketch' ensure you select your Arduino Uno in the drop-down box at the top left side of the tool bar. To upload a sketch to your Arduino you will need to first verify it by clicking the check mark at the top of the tool bar and then uploading it by clicking the arrow.

6. Code Tutorial

The first thing that must be accomplished is defining two variables: a 'time interval' of int type and a 'previous time' of int type as well. We set our 'timeInterval' variable to 2000, which means our interval is 2 seconds.

```
const int timeInterval = 2000; //The time interval at which we will print in mS.
int previousTime = 0; //This will hold the previous value of time once a loop iteration has
                      //completed.
```

Next, in our `void setup` function, we define code that we want to run only once. In the typical case with an Arduino, we will define our baud rate of 9600. In our context of serial ports, which is how the Arduino communicates with your computer, 9600 baud simply means that it is capable of transferring 9600 bits per second.

```
void setup() {
  //This contains actions we will take prior to executing our main loop. In this case, we set
  //our baud rate to 9600 - standard.
  Serial.begin(9600);
}
```

Finally, we can setup our `void loop` function. This function contains our primary block of code which will loop repeatedly while the Arduino is powered on. The process is as follows:

1. The loop will update a int variable ‘currentTime’ with a built-in millis function.
2. An if statement will check if the following logic statement is true: is the current time minus the previous time greater than or equal to the defined time interval. If so, we will print our defined statement via serial print, which in our case is “Hello World”. Finally, we update the value of ‘previousTime’ to be that of ‘currentTime’. The loop will then continue for as int as we allow the Arduino to remain powered on.

```
void loop() {
  /* At the function loops, it will constantly be updating currentTime
     until the condition of our if statement is met. When it is met it
     will print our requested line and then set and save our current time
     to previous  time. This allows us to keep track of how long we have
     been printing.
   */
  unsigned int currentTime = millis();
  if(currentTime - previousTime >= timeInterval) {
    Serial.println("Hello World");
    previousTime = currentTime;
    }
}
```

Note: In order to see the serial monitor, select it under “Tools” in the tool bar or press “Crtl + Shift + M”.

The final output should look something like this:

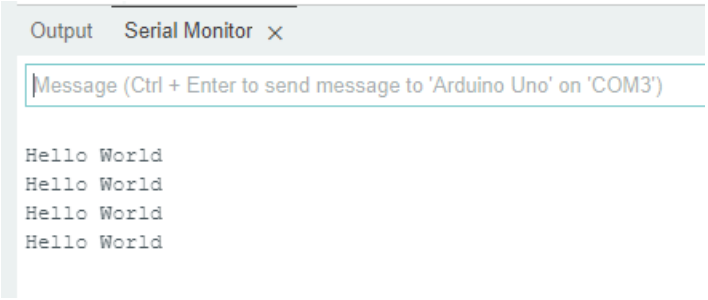


Figure 2: Output from Serial Monitor

7. Further Exercises

Now that you have taken on the basics of this project, here are a few challenges to accomplish that will help you build on your skills.

1. Change the interval time to a much smaller value, such as 1 ms. Why is this not ideal?
2. Why is the `currentTime` variable in the loop function an unsigned integer? What are the advantages of using this variation?
3. In the `void loop` function, add an integer which incremented on by 10 every 10 loops in place of our current print statement to keep track of how many loop iterations have been completed. What were some considerations you took into account while accomplishing this?