

CS 428 Lecture Notes

06 March 2019

In the Arduino IDE, the Serial Monitor can be found on the top right corner. If you want to display data, you can do so via the serial monitor. To get something to print out of the serial monitor, you can type `serial.print.println()`. Before that, you must type `serial.begin(9600)` in setup. Numbers can be created in the global space. Below is a screenshot of a working version of the program

Figure 1: Multiplying by 2 with Arduino

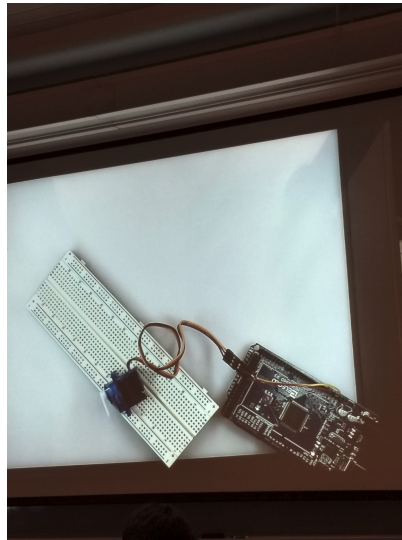
```
long number=0;
long a=0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  number = 0;
  Serial.flush();
  while(Serial.available()==0)
  {
    \\no incoming stream of inputs via serial_monitor
  }
  while(Serial.available(>0)
  {
    a = Serial.read()-'0'; //subtracting by the ascii value of 0, which is 48 to get the actual number
    if(a==38) continue; //get a new input. newline was inputted
    number=number*10;
    number=number*a;
    delay(5); //delay is in miliseconds. is necessary for code to run correctly
  }
  Serial.print("You entered: "); //prints without the newline character
  Serial.println(number); //prints with newline character
  Serial.print(number);
  Serial.print(" multiplied by two is: ");
  number = number*2;
  Serial.println(number);
  Serial.flush();
}
```

`Serial.available()` checks for an available stream.
If what is in the parentheses is empty, `System.available()` returns 0.
`Serial.flush()` empties the memory, which is good for future iterations between what we have running, which is a two-way stream between a computer and an Arduino.
`Delay` is used as a safeguard against glitches (safety protocol), and makes the stream stable. We can use the Control Board and the Servo Motor (and some wires), we can measure the the temperature around the Servo Motor.
Below is an image of the setup (I apologize for the relative lack of clarity).

Figure 2: Temperature Measurement setup



We can either use a different setup, but doing so would require the installation of a particular library. Instead of that, we can use a pre-built library (pictured below).

Figure 3: Servo Code

```
#include <Servo.h>

Servo myservo;

void setup(){
  // put your setup code here, to run once:
  myservo.attach(4); //pin number
}

void loop() {
  // put your main code here, to run repeatedly:
  myservo.write(100); //sets speed, causing piece on servo to rotate at the angle specified
  //in this case 100 deg
  delay(1000); //in milliseconds
  myservo.write(90);
  delay(1000);
  myservo.write(80);
  delay(1000);
}
```

The servo also has a part that a head can connect to, but the head cannot move at a 360 degree angle. The delay delays the servo, from spinning continuously.

servo is good for moving an object periodically.

The pin number is the pin number in which the signal is attached to (in case my comment on the code wasn't clear enough).