# EE – 351 (Control & Robotics Laboratory)

## Experiment – 3

Vidhi Sahai

190102085

**Abstract**

In this module we need to design a simple sensor-network. To achieve the task at hand we learn the use of the analog input pins and Analog-to-Digital (ADC) inside an Arduino. We further learn to interface a servo motor to an Arduino. The sensor-network to be designed will help us automate Google's dino game. The dino game can be accessed using the following link: chrome://dino. Your objective for this module is to design a system that automates playing this game.

Answer the following:

1. Is the designed system an open-loop system or a closed-loop system? Depending on your answer map the components of your system to the different building blocks of an open-loop/ closed-loop system.

2. On executing the command servo.write(30) in Arduino Uno, a servomotor connected to the suitable pin of the Arduino rotates 30°. How does the servomotor rotate to the designated angle (Explain the internal working principle)? Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP (In-cicuit serial programming) header and a reset button. Embedded C is a set of language extensions for the C programming language by the C Standards Committee to address commonality issues that exist between C extensions for different embedded systems. Whenever the conventional C language and its extensions are used for programming embedded systems, it is referred to as "Embedded C" programming. We will be using the Arduino Uno R3 board along with a breadboard and some other basic electrical components to build all our circuit modules. We will be programming our Arduino Uno R3 using Embedded C programming language.

## Module – 3 (Automated Dino Game Using Arduino)

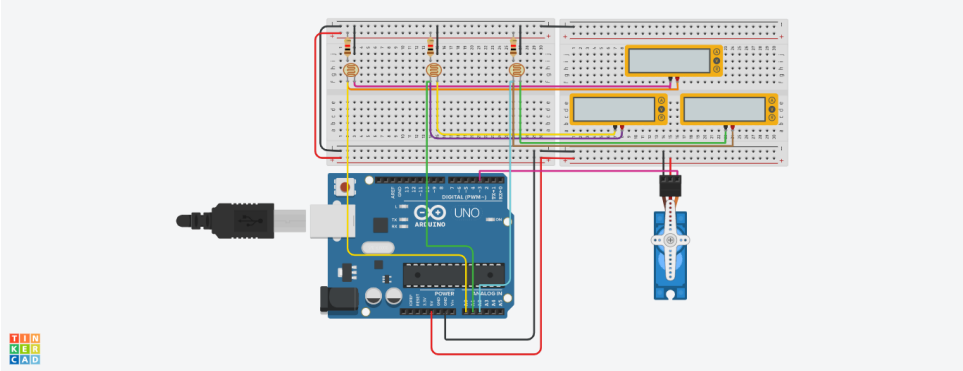
Main code

```
// C++ code

//



// include the Servo.h library

#include<Servo.h>



Servo servo; // instantiate a servo object to control the servo



int val1, val2, val3; // variables to store the voltages across the photoresistors

int pos = 0; // variable to store the position of servo
```

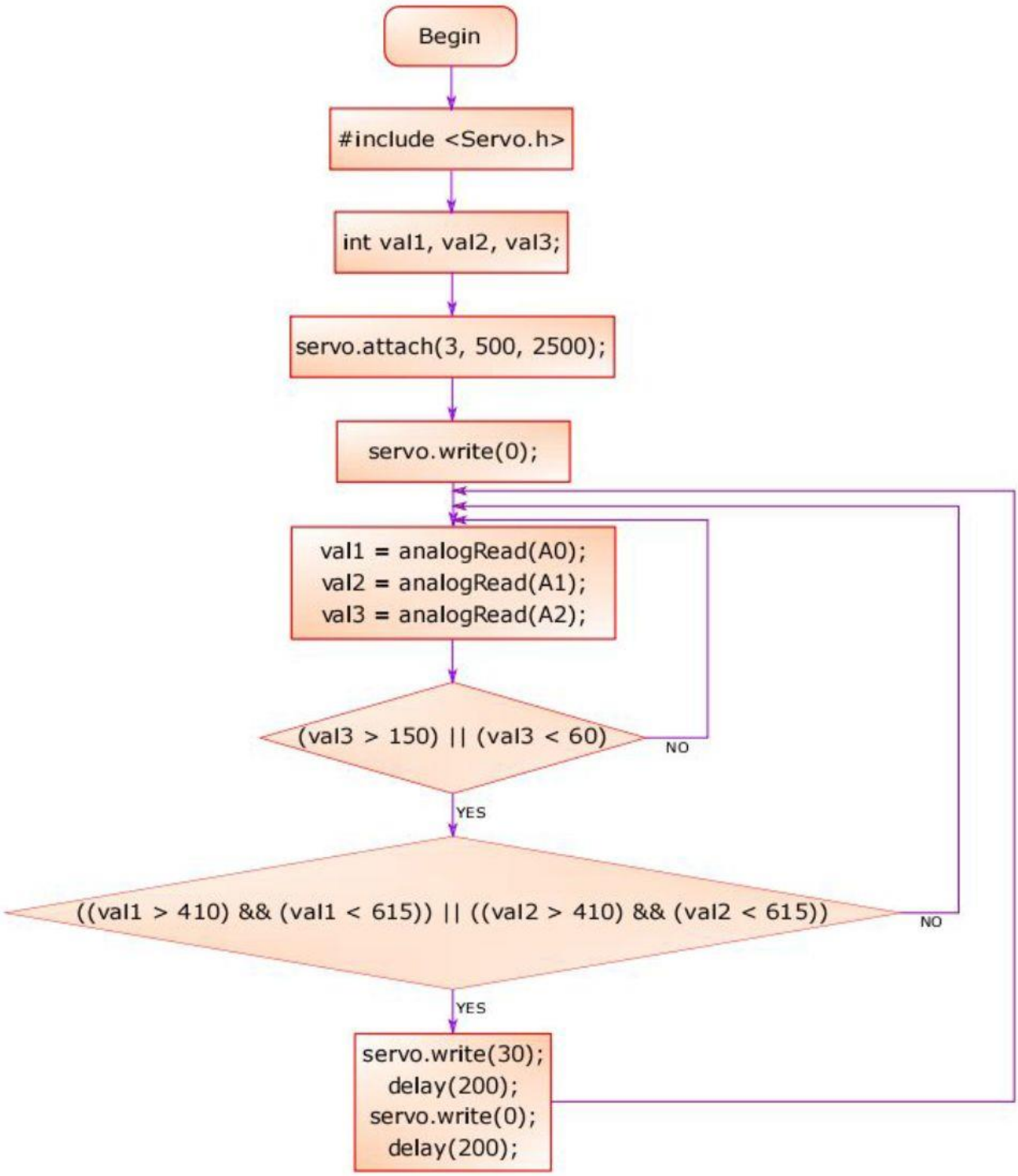
```
void setup()
{
  servo.attach(3, 500, 2500); // attaches the servo on pin 3 to the servo object
  servo.write(0); // initialize servo to start at 0 degree position
}

void loop()
{
  // read photoresistor voltages
  val1 = analogRead(A0);
  val2 = analogRead(A1);
  val3 = analogRead(A2);

  // light background
  if (val3 > 150)
  {
    if (((val1 > 410) && (val1 < 615)) || ((val2 > 410) && (val2 < 615)))
    {
      servo.write(30); // turn servo to 30 degrees
      delay(200); // wait 200 ms for it to have turned
      servo.write(0); // turn servo to 0 degrees
      delay(200); // wait 200 ms for it to have turned
    }
  }

  // dark background
  else if (val3 < 60)
```

```
 {
  if (((val1 > 410) && (val1 < 615)) || ((val2 > 410) && (val2 < 615)))
  {
   servo.write(30); // turn servo to 30 degrees
   delay(200); // wait 200 ms for it to have turned
   servo.write(0); // turn servo to 0 degrees
   delay(200); // wait 200 ms for it to have turned
  }
 }
}
```

## Circuit Diagram



## Flowchart




**Observations**

When we run the simulation, we see the all 3 sensors are set to dark. If we move either LDR1 or LDR2 through the grey region, we see that the servo rotates by 30° and then comes back to 0°. This happens if we keep LDR3 either in white region or in black region. If we keep LDR3 in the grey region, then regardless of the regions in which LDR1 and LDR2 are, the servo will remain stationary. Hence, our automation works as expected.

## Answers to Questions

*1) Is the Designed System an Open-Loop System or a Closed-Loop System?*

The designed system is a closed-loop system. The circuitry in the servo driver compares the tachometer feedback voltage and determines if the desired speed has been accomplished - known as a closed velocity loop. The velocity loop is monitoring the commanded velocity and tachometer feedback, while the driver adjusts the power to the motor to maintain the desired commanded velocity. In a more sophisticated servo motor system, multiple embedded loops are tuned for optimal performance to provide precision motion control. The system consists of current, velocity, and position loops that utilize precision feedback elements. Each loop signals the subsequent loop and monitors the appropriate feedback elements to make real time corrections to match the commanded parameters.

*2) How Does the Servomotor Rotate to the Designated Angle?*

A servo motor controls the rotation of a DC motor through a control circuit that adjusts its angle. Control is achieved by adjusting the length of a square wave pulse sent to the servo motor. The length of the pulse in a train of signals is defined by the Pulse Modulation Width (PMW). Depending on the duty cycle, the angle of rotation will vary. Starting from a duty cycle of 1%, corresponding to the angle of 0°, as the duty cycle gradually increases, the angle of rotation of the servo motor will also increase. With a 50% duty cycle, the motor pin will position itself at 90° (which is just half of the possible rotation range). With a duty cycle of

99%, we will have a rotation of 180°. Then, through this PMW modulation system, we will be able to send commands from the outside to the servo motor to make it assume the desired angle of rotation.

## References


[1] https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode/.

[2] https://store.arduino.cc/usa/arduino-uno-rev3.

[3] https://www.arduino.cc/reference/en.

[4] https://www.kollmorgen.com/en-us/blogs/_blog-in-motion/articles/how-servo-motors-work/.

[5] https://www.meccanismocomplesso.org/en/arduino-servo-motors-how-they-work-and-how-to-control-them/.