Fabian Neumüller, Constantin Gemmingen, Luke Peppard - Group 4.3

PreLab04

1. **What is the range of PWM pulse lengths that can be used to set the position of the Futuba s3003 servo?**

From 550 microseconds to 2300 microseconds

2. **What PWM pulse length do you need to set to position the servo at 66.6°?**

2’300 - 550 = 1’750 microseconds → 180° = 1’750 microseconds → 1° = 9.722 microseconds → 66.6° = 647.5 microseconds
⇒ total pulse length: 647.5 + 550 = 1197.5 microseconds

3. **What is the fastest that the servo can run through its full, 180° range of motion?**

Fastest at 6.0 V is 0.57 seconds (0.19 sec/60°)

4. **Is this s3003 servo a good or a bad choice for a clock? Explain.**

Clock Signal (Telecommunication): From the datasheet, we only know the pulse length (550-2300 microseconds) but we don’t know the total amount of time it takes for the next pulse to arrive so we can’t calculate how many cycles it would take for one second to pass. If we knew the constant time for one full pulse (incl. pause afterwards), we could calculate how many full pulses we would have to wait for one second to pass and could build a clock that way. Given that the time period between pulses is constant and known, the servo would make a good clock.
Wall-mounted-Clock: considering the high angular resolution and high positional precision of a servo it could move the hands on a clock precisely. The required precision is 6° per minute (360°/60 min).

5. **List a different application that you think could use the high position accuracy of servos (that has not yet been listed in the introduction: RC planes and toy cars, opening and closing DVD disc trays, grippers in robots).**

- Swiping left on tinder
- closing/opening valves
- Automatic door openers

- Camera auto focus

6. **What is the result of increasing the input voltage of the servo from 5V to 6V in terms of speed and torque?**

The speed is increased from 0.23 sec/60° to 0.19 sec/60° while the torque increases from 31.1 Ncm to 40.1 Ncm

7. **How much does a single s3003 cost approximately?**

CHF 15.95 (Digitec), US $2.69 (Alibaba)

8. **Take the setup of Figure 4.3. Assume Gear A has 60 teeth, Gear B has 18 teeth, Gear C has 48 teeth, Gear A rotates at 5 RPM and Gear D rotates at 40 RPM. How many teeth does Gear D have?**

$$\frac{z_B}{z_A} * \frac{z_D}{z_C} = r = \frac{\omega_A}{\omega_D} \longrightarrow \frac{18}{60} * \frac{z_D}{48} = \frac{5}{40} \longrightarrow z_D = 20$$

⇒ gear D has 20 teeth

9. **Read the documentation on the Arduino servo library servo.h. You can find it by following this link: https://www.arduino.cc/en/reference/servo. In your own words, explain what the 6 functions of the library do and what their in- and outputs are. Write at most 4 sentences per function.**

- `attach(pin, min, max)`: tell the Arduino, to which pin the servo motor is attached and what the pulse widths (in microseconds) are that translate from the minimum and maximum angles. specifications: You can only attach the pins 9 or 10; min and max are optional and 544 and 2400 microseconds are chosen respectively, per default
- `write(angle)`: angle ranges from 0 to 180. You can set the angle of the shaft of the servo to the desired position using this function. The value of “angle” translates directly to the position.
- `writeMicroseconds(uS)`: gives a value in microseconds to the servo and thereby controls the position of the servo’s shaft. A value of 1000 corresponds to a fully counterclockwise position and a value of 2000 to a fully clockwise and therefore a value of 1500 corresponds to the middle position for standard servos.
- `read()`: reads the current angle of the serve (between 0 and 180 degrees)

- `attached()`: checks if a servo is attached to a pin and returns "true" if there is one attached, returns "false" if none is attached.
- `detach()`: clears the pins that might have been blocked by the `attached()` function

10. **Solve PostLab Q1. You can test your code by using the already assembled exercise on TinkerCAD: https://www.tinkercad.com/things/fnHdZKLGPtf Make an Autodesk account if you don't already have one, click on "Tinker This". On the top right you will see several buttons. To edit the code on the Arduino press on "Code". To start the simulation press on "Start Simulation" (Figure 4.6). If you correctly solve the exercise you will see the change in the width of the PWM pulse on the Oscilloscope. Feel free to play around some more with TinkerCAD and test other electronic components.**
    a. Note: The Arduino that you can use on TinkerCAD is an Arduino UNO. There are several differences to the Arduino DUE we previously used in the lab: Its Analog Input Pins accept voltages from 0 to 5V instead of 0 to 3.3V. Keep that in mind when you work with the circuit. The analog read resolution of the Arduino UNO is 10 bit and can not be set to 12 bit, as with the Arduino DUE. Additionally, the Servo Motor needs 5V to operate, contrarily to the one we would have used in the lab.

PostLab Q1:
TinkerCAD: https://polybox.ethz.ch/index.php/s/gncN632mTBOVOwG
Tested on:

```
#include <Servo.h>

// declare variables

#define poti_pin A0  // analog pin used to connect the potentiometer
float val;    // variable to read the value from the analog pin
float minPoti = 0;
float maxPoti = 1024; // 1024 is the resolution of the potimeter
int angle;

// create a servo object to control the servo
```

Fabian Neumüller, Constantin Gemmingen, Luke Peppard - Group 4.3

```
Servo lukesServo;
int minPulse = 550;
int maxPulse = 2300;
int minAngle = 0;
int maxAngle = 180;
void setup() {

  // _______________ Begin - Setup _______________

  // Attach the servo on pin 9 to the servo object
  lukesServo.attach(9, minPulse, maxPulse);
  // Begin the serial communication
  Serial.begin(115200);

  // _______________ End - Setup _______________

}

void loop() {
  // _______________ Begin - Loop _______________

  // Read values from the analog pin and map/scale them to the movment range of
the servo.
  val = analogRead(poti_pin);
  angle = map(val, minPoti, maxPoti, minAngle, maxAngle);
  // Display the reading from the potentiometer on the serial monitor

  // Serial.print(angle);
  // Serial.print ("//");
  Serial.println(val);
  // Set the servo position according to the scaled value
  lukesServo.write(angle);

  // _______________ End - Loop _______________

  delay(15); // wait for the servo to get there
}
```