Engineering Summer Camp for the Elite 2016

Department of Computer Science

Arduino Programming

Introduction

Arduino is a company that designs and manufactures the open-source hardware and software, microcontroller-based kits that can be used in many different places. In this section, we are going to write some simple programs to control our robot.

Step 1. Understanding our first program

The program below is the one we have used in setting up the robot the first time.

```
                                                    // reset.ino
#include <Servo.h>

#define START_PIN 2

Servo servo[8];

void setup() {
  int i;
  for (i = 0; i < 8; i++) {
    servo[i].attach(i + START_PIN);
    servo[i].write(90);
  }
}

void loop() {
}
```

We first declare that we need to use Servo in our program, by including the Servo library.

```
#include <Servo.h>
```

We define a constant, this is our starting pin number (remember that we connect our first servo at pin D2?).

```
#define START_PIN 2
```

We define 8 servos in our program.

```
Servo servo[8];
```

We define the **`setup()`** function. This function will be executed once by Arduino board to initialize the board.

```
void setup() {
…
}
```

We define a for-loop. It uses a variable **`i`** and execute everything inside with the value of **`i`** changing from 0 up to 7. Here, **`i = 0`** set **`i`** to 0 before the loop; **`i < 8`** is the condition of this loop, and **`i++`** means that after each iteration of the loop, **`i`** will be incremented by 1.

```
int i;
for (i = 0; i < 8; i++) {
…
}
```

Inside the loop, we set up the relationship of the "servo" in the program and the "servo" in the robot by specifying the corresponding pin number. For example, when **`i`** is 0, it attach **`servo[0]`** to pin 2, i.e., leg 0 servo A.

```
servo[i].attach(i + START_PIN);
```

Still inside the loop, we immediately set the servo to 90 degree.

```
servo[i].write(90);
```

We define the **`loop()`** function. This function will be executed repeatedly by Arduino board after setup. We leave this function empty at the moment.

```
void loop() {
}
```

Step 2. Adjusting the servos for the initial position

When all servos are set to 90 degree, the legs should return to the "**initial**" state. Depending on how well the servos are installed, the position of the legs may not be consistent. There are two ways to fix this:

1. Disassemble part of the robot to adjust the servo. This is the better method as it allows you to share your code with others.
2. Define a set of offset values to the servos, which adjust the angle to be set. This is easier as there is no need to disassemble the robot. We will use this method.

Let's modify our program.

```
                                                    // offsets.ino
#include <Servo.h>

#define START_PIN 2

Servo servo[8];
```

```
int offsets[8];

void setup() {
  int i;
  offsets[0] = 0;
  offsets[1] = 0;
  offsets[2] = 0;
  offsets[3] = 0;
  offsets[4] = 0;
  offsets[5] = 0;
  offsets[6] = 0;
  offsets[7] = 0;
  for (i = 0; i < 8; i++) {
    servo[i].attach(i + START_PIN);
    servo[i].write(90 + offsets[i]);
  }
}

void loop() {
}
```

We have added a few things. First we define the offset variable together with the servos.

```
int offsets[8];
```

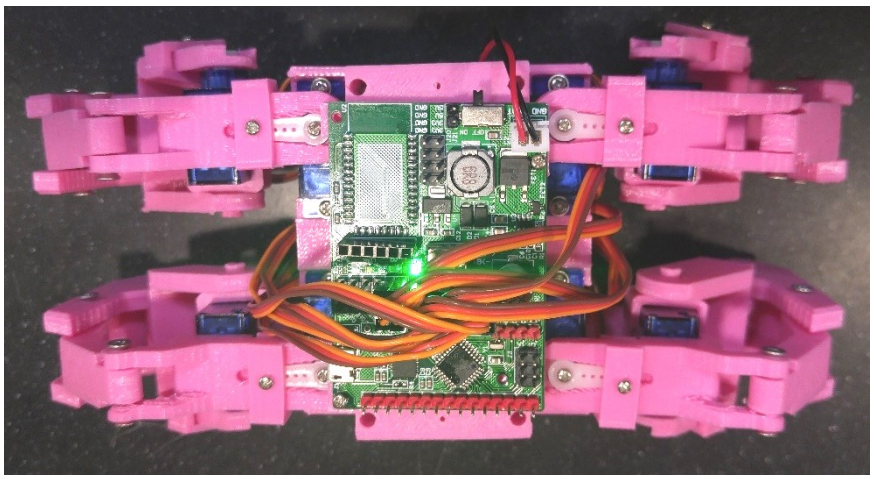
Then we set up these offsets, at this point we will just set them to zero.

```
offsets[0] = 0;
offsets[1] = 0;
…
```

Finally instead of simply setting the servo angles to 90, we add an offset to it.

```
servo[i].write(90 + offsets[i]);
```

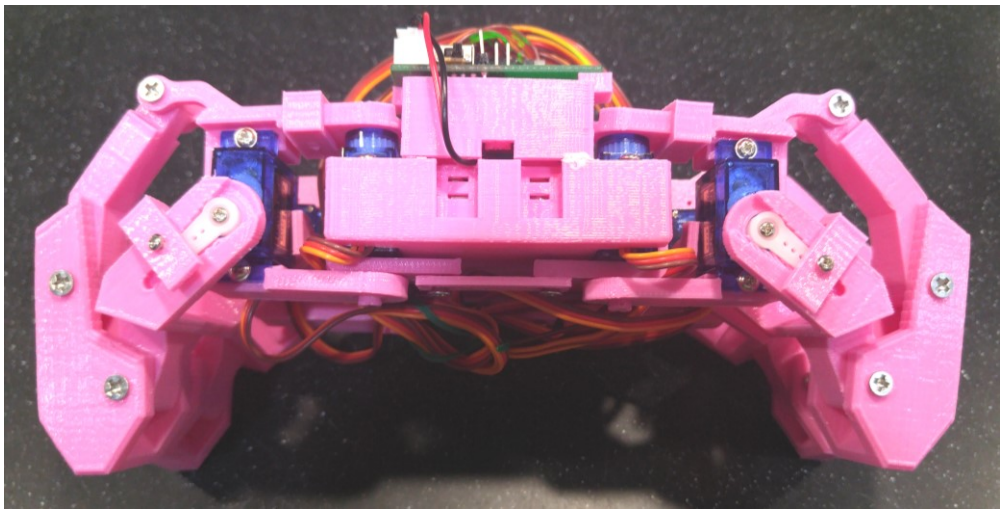
Now we can adjust the values of these offsets, so that when the robot is switched on, it will be its best initial state (legs perpendicular to body, legs in the downward position, well balanced).



Top-view of initial position

To do so, adjust the offset values one by one and test it out on your robot. Start by setting a value of 10 or -10 and increment/decrement slowly (you can do it in a step of +5/-5).

Please note that the downward position should **not** be the point where the leg extends to its very limit.



Side-view of showing the leg-down position

Check point 1: Ask for a checking before you continue.

Step 3. Finding the range of movement

The initial state of the robot marks one end of the movement range. Our next program helps us to find the other end. We start from our last program, with the offset values found.

```
                                                    // stretching.ino
…
void setup() {
  …
  for (i = 0; i < 8; i++) {
    servo[i].attach(i + START_PIN);
//  servo[i].write(90 + offsets[i]);
  }
  servo[0].write(90 + offset[0]);
  servo[1].write(90 + offset[1]);
  servo[2].write(90 + offset[2]);
  servo[3].write(90 + offset[3]);
  servo[4].write(90 + offset[4]);
  servo[5].write(90 + offset[5]);
  servo[6].write(90 + offset[6]);
  servo[7].write(90 + offset[7]);
}
…
```

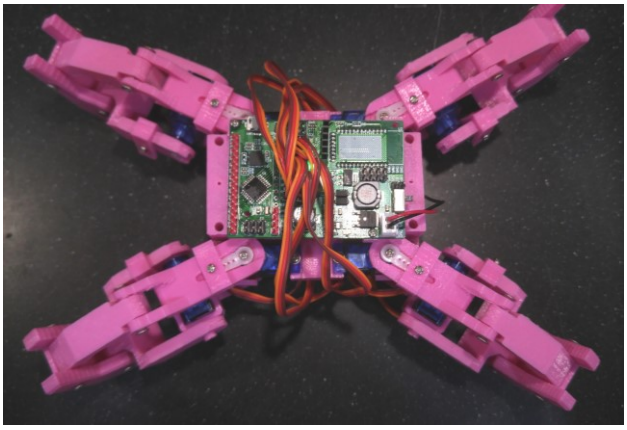
We removed the line setting all servo angles to 90.

```
servo[i].write(90 + offsets[i]);
```

Then we set the angles one by one.

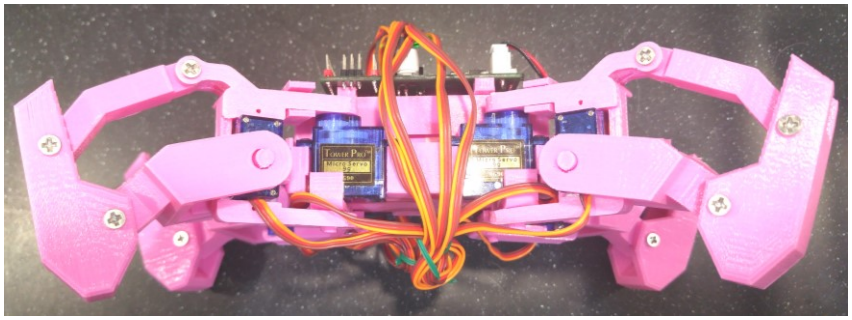
```
servo[0].write(90 + offset[0]);
servo[1].write(90 + offset[1]);
…
```

Now, instead of setting them to 90, we will change these to some other value so that all four legs are moved forward/backward. Let's call this position the alternate position. Do not move to the limit, 10 degrees before reaching the end will be enough.



Top-view of the robot in the alternate position

Similarly, move the leg to the upward position so that the body of the robot is lowered. You should be able to slide your finger under the robot easily under this state.



Now record your result in the table below. Ideally you should see some patterns in this table.

Leg	Servo	servo [?]	Initial position	Offset	Alternate position	Servo	servo[?]	Leg down	Offset	Leg up
0	A	**servo[0]**	90			B	**servo[1]**	90		
1	A	**servo[2]**	90			B	**servo[3]**	90		
2	A	**servo[4]**	90			B	**servo[5]**	90		
3	A	**servo[6]**	90			B	**servo[7]**	90		

Check point 2: Ask for a checking before you continue.

Step 4. Playing with the loop

With the previous results, we can put our robot in motion now. First we try to move one leg.

```
                                                        // oneleg.ino
…
void setup() {
  int i;
  for (i = 0; i < 8; i++) {
    servo[i].attach(i + START_PIN);
    servo[i].write(90 + offsets[i]);
  }
  delay(3000);
}

void loop() {
  // add your code to move leg 0 to alternate position, leg up
  delay(300);
  // add your code to move leg 0 to the initial position, leg down
  delay(300);
}
```

We first add a delay at the end of **`setup()`**. This allow us some time to put down the robot after switching it on. The number is in milliseconds, i.e., **`delay(3000)`** will delay for 3 seconds.

```
delay(3000);
```

The **`loop()`** function defines one cycle of action. The Arduino board will execute it repeatedly. As the servos need time to move to a position, you should always introduce a delay after setting a position.

At this point you should know how to move a leg to the initial/alternative position. So do it here in the place indicated.

```
// add your code to move leg 0 to alternate position, leg up
delay(300);
// add your code to move leg 0 to the initial position, leg down
delay(300);
```

Challenge: Spinning robot

Modify your loop to do this sequence of action in one cycle (remember to add delay after each action):

1. Leg 0 up
2. Leg 0 moves to alternate position, leg down
3. Leg 1 up
4. Leg 1 moves to initial position, leg down
5. Leg 2 up
6. Leg 2 moves to alternate position, leg down
7. Leg 3 up
8. Leg 3 moves to initial position, leg down
9. Leg 0 and 2 move to initial position, leg 1 and 3 move to alternate position.