Engineering Summer Camp for the Elite 2016

Department of Computer Science

Dancing robot

Introduction

This guide is also available at https://goo.gl/vXnVpF .

Before the racing competition, there will be a free-style demo session. During the session, you can show to the guest some great moves of your robot.

Step 1. Defining moves

The easiest way to robot dancing is to define a number of sequences as functions, so that the **`loop()`** function will perform one of the sequences at a time.

Here is an example of how it can be done.

```
                                                    // multiple.ino
…
#define MOVES 4
void action(int);

void left_spin();
void right_spin();
void some_cool_moves();
void some_more_cool_moves();

…

void action(int move) {
  if (move == 0) {
    left_spin();
  } else if (move == 1) {
    right_spin();
  } else if (move == 2) {
    some_cool_moves();
  } else if (move == 3) {
    some_more_cool_moves();
  }
}

void left_spin() {
…
}
…
```

We first define the number of moves.

```
#define MOVES 4
```

Then declare a function that translate a value to a move.

```
void action(int);
```

Followed by the set of functions that perform the moves.

```
void left_spin();
void right_spin();
void some_cool_moves();
void some_more_cool_moves();
```

At the end of the code, we can define the functions. First the function that translate a value to a move. Then followed by the functions that perform the moves.

```
void action(int move) {
  if (move == 0) {
    left_spin();
  } else if (leg == 1) {
    right_spin();
  } else if (leg == 2) {
    some_cool_moves();
  } else if (leg == 3) {
    some_more_cool_moves();
  }
}
```

Step 2. Cycling through the moves

The first way to use the moves one by one repeatedly.

```
                                                    // multiple.ino
…
int m = 0;
void loop() {
  action(m);
  m = (m + 1) % MOVES;
}
…
```

First define variable **m** outside of the function. We need to define it outside as we need to use it across all execution of the **loop()** function.

```
int m = 0;
```

Inside **loop()**, we use the **action()** function we have just defined to perform a move.

```
action(m);
```

After that, we change **m** to the next move.

```
m = (m + 1) % MOVES;
```

Followed by the set of functions that perform the moves.

Step 3. Performing random moves

The second way to use the moves is to randomly pick a move every time.

```
                                                    // random.ino
…
void setup() {
  …
  randomSeed(analogRead(0));
}

void loop() {
  int m = (int)random(MOVES);
  action(m);
}
…
```

First initialize the random number generator in **setup()**.

```
randomSeed(analogRead(0));
```

Inside **loop()**, we use the **random()** function to get a random number. In our case, **random()** will return a value in the range of $[0, 4)$. (i.e. from 0 to 3, a total of 4 possible values)

```
int m = (int)random(MOVES);
```

Then we can perform the move accordingly.

```
action(m);
```

Followed by the set of functions that perform the moves.

Step 3. Performing timed moves

We can also repeat the same move for some amount of time before changing to the next move.

```
                                                                    // timed.ino
…
#define DURATION 5000
int last_time = 0;
int m = 0;
…
void loop() {
  int curr_time = millis();
  if (last_time - curr_time > DURATION) {
    last_time = curr_time;
    // change move
  }
  action(m);
}
…
```

First define a constant for the duration of each move.

```
#define DURATION 5000;
```

Setup two variables, one to keep the last time we have changed a move, the other to keep the current move.

```
int last_time = 0;
int m = 0;
```

Inside the **`loop()`** function, we first get the current time.

```
int curr_time = millis();
```

Check if this is the time we change to the next move. If that is the case, we update the time, and change move (either cycle to the next move or by random).

```
if (last_time - curr_time > DURAION) {
    last_time = curr_time;
    // change move
}
```

Finally before leaving the loop, perform the action.

```
action(m);
```

You are on your own now, be creative!