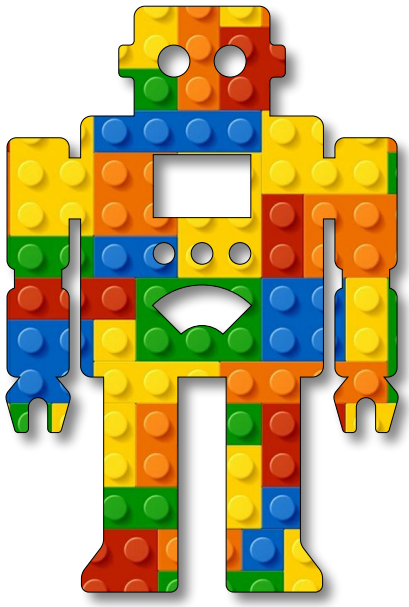
Intelligent Machines Module A Unit #4 functions



Content

Module A Unit #4 Functions

Introduction to functions

As you know, functions are a key part of the coding world. This is also true with robotics. This unit takes you through the basics and the syntax of functions in C/C++. It is very similar to p5.js, and so the learning curve is not too steep.

Definition: Segmenting code into functions allows a programmer to create modular pieces of code that perform a defined task and then return to the area of code from which the function was "called". The typical case for creating a function is when one needs to perform the same action multiple times in a programme.

Sketch A4.1 a blinking function

We have a function called `void setup()`, a function called `void loop()`, and now we are going to create one called `void blink()`. This is a made-up function where we are going to put our code for blinking.

Arduino sketch

```
int delayPeriod = 500;

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 10; i++)
  {
    digitalWrite(13, HIGH);
    delay(delayPeriod);
    digitalWrite(13, LOW);
    delay(delayPeriod);
  }
  delay(1000);
}

void blink()
{

}
```

🗒️ Notes

We can put the function anywhere, but it does start at the beginning and work its way down.

🛠️ Code Explanation

`void blink()`	We have created a new function called blink()

Sketch A4.2 filling the blink

Now we are going to call it from inside the `for()` loop. This is a cut-and-paste job, taking the code out and replacing it with the `blink()` function.

Arduino sketch

```
int delayPeriod = 500;

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 10; i++)
  {
    blink();
  }
  delay(1000);
}

void blink()
{
  digitalWrite(13, HIGH);
  delay(delayPeriod);
  digitalWrite(13, LOW);
  delay(delayPeriod);
}
```

Notes

This can be useful for keeping the code neat and tidy, especially where you want to use the same bit of code repeatedly. One of the targets that coders try to achieve is to write as few lines as possible to achieve the same result. It is considered bad form to repeat the same lines of code.

Code Explanation

`blink();`	This blink() function is called ten times in the for() loop

Sketch A4.3 is this an argument?

Here we will combine functions and arguments. Using the above sketch, we can rationalise it even more by using the two arguments in the function itself. We will start with just one and then add the other afterwards.

❗ Remove the line of code: `int delayPeriod = 500;`

Arduino sketch

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 10; i++)
  {
    blink(500);
  }
  delay(1000);
}

void blink(int delayPeriod)
{
  digitalWrite(13, HIGH);
  delay(delayPeriod);
  digitalWrite(13, LOW);
  delay(delayPeriod);
}
```

🛠 Code Explanation

`void blink(int delayPeriod)`	This is a variable inside a function, you can call it an argument. It will hold that value when it is called from somewhere else, in this case 250

Sketch A4.4 learning to count

We can go one step further by adding in the `count` value of `10`. There is a bit of cutting and pasting, so work through the sketch to see how it is doing what it does.

Arduino sketch

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  blink(500, 10);
  delay(1000);
}

void blink(int delayPeriod, int count)
{
  for (int i = 0; i < count; i++)
  {
    digitalWrite(13, HIGH);
    delay(delayPeriod);
    digitalWrite(13, LOW);
    delay(delayPeriod);
  }
}
```

🗒️ Notes

You have initialised them inside the function brackets. You draw their values from the `void loop()` function. We create an integer variable called `count`.

🌻 Challenge

Change the values in `blink(500, 10);`

🛠️ Code Explanation

`int delayPeriod`	Both are variables that are arguments. They each take on the values of 500 and 10 respectively
`int count`	

Sketch A4.5 a blinking stop

❗ This is our new starting sketch.
In this next sketch, we will use the += operator to slow the blink down.

Arduino sketch

```
int delayPeriod = 1000;


void setup()
{
  pinMode(13, OUTPUT);
}


void loop()
{
  digitalWrite(13, HIGH);
  delay(delayPeriod);
  digitalWrite(13, LOW);
  delay(delayPeriod);
}
```

Sketch A4.6 increasing blink

Changing the `delayPeriod` to something much shorter, `20` milliseconds, and increasing it by `10` milliseconds on each blink.

Arduino sketch

```
int delayPeriod = 20;

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(delayPeriod);
  digitalWrite(13, LOW);
  delay(delayPeriod);
  delayPeriod += 10;
}
```

🛠️ Code Explanation

`delayPeriod += 10;`	We add 10 to the variable delayPeriod on every iteration

Sketch A4.7 stop blinking

If we wanted to stop the blinking at 1000 milliseconds, we would need to first introduce another variable. We will call this increment and make it much bigger, i.e. 50.

Arduino sketch

```
int delayPeriod = 20;
int increment = 50;

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(delayPeriod);
  digitalWrite(13, LOW);
  delay(delayPeriod);
  delayPeriod += increment;
}
```

Sketch A4.8 and stop

Stopping at 1 second (1000 milliseconds).

Arduino sketch

```
int delayPeriod = 20;
int increment = 50;

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(delayPeriod);
  digitalWrite(13, LOW);
  delay(delayPeriod);
  delayPeriod += increment;
  if (delayPeriod > 1000)
  {
    increment = 0;
  }
}
```