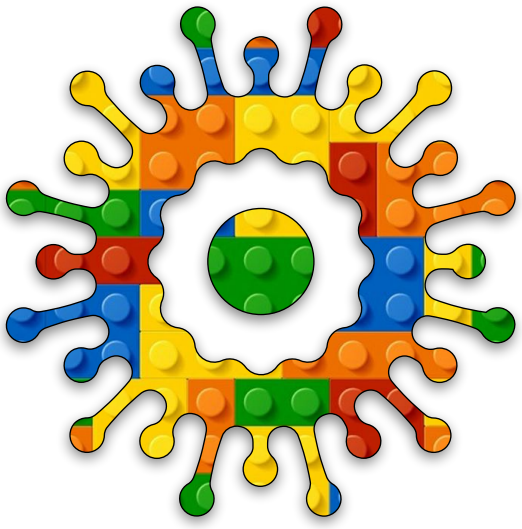
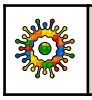
Internet of Things

Module A

Unit #4

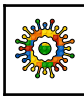
functions





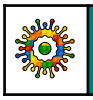
Contents

Module A Unit #4 functions



Introduction to functions

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 void blink()

We have a function called `void setup()`, and 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.

```
int delayPeriod = 250;

void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 10; i++)
  {
    digitalWrite(2, HIGH);
    delay(delayPeriod);
    digitalWrite(2, LOW);
    delay(delayPeriod);
  }
  delay(2000);
}

void blink()
{

}
```

🗒️ Notes

The LED will blink very fast, 10 times, stop, and then repeat. The new function doesn't do anything yet. We can put the function anywhere, but it does start at the beginning and work its way down the lines of code.

🛠️ Code Explanation

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

Sketch A4.2 the blink function

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.

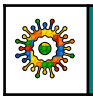
```
int delayPeriod = 250;

void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 10; i++)
  {
    blink();
  }
  delay(2000);
}

void blink()
{
  digitalWrite(2, HIGH);
  delay(delayPeriod);
  digitalWrite(2, 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 lines of code unnecessarily.

🛠️ Code Explanation

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



Sketch A4.3 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. We will also change the `delayPeriod` to `100`.

❗ Remove the line of code: `int delayPeriod = 250;`

```
// int delayPeriod = 250;

void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 10; i++)
  {
    blink(100);
  }
  delay(2000);
}

void blink(int delayPeriod)
{
  digitalWrite(2, HIGH);
  delay(delayPeriod);
  digitalWrite(2, LOW);
  delay(delayPeriod);
}
```

🗒️ Notes

You will notice that it blinks much faster (or should).

🛠️ Code Explanation

`blink(100);`	We have give the function an argument
`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 100

Sketch A4.4 counting to 10

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 and why it is doing exactly what it does. We will change the rate of blinking so you can see a change.

❗ Remove the `for()` loop from `void loop()` and put it in `void blink()` if in doubt just retype the whole thing, it is good practice anyway!

```
void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  blink(100, 10);
  delay(2000);
}

void blink(int delayPeriod, int count)
{
  for (int i = 0; i < count; i++)
  {
    digitalWrite(2, HIGH);
    delay(delayPeriod);
    digitalWrite(2, LOW);
    delay(delayPeriod);
  }
}
```

🗒️ Notes

You have initialised them inside the function brackets. You draw their values from the `void loop()` function.

🌻 Challenge

Change the values in `blink(100, 10);`

🛠️ Code Explanation

`void blink(int delayPeriod, int count)`	Both are variables (`delayPeriod` and `count`) that are also arguments. They each take on the values of 100 and 10 respectively

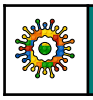
Sketch A4.5 a blinking stop

❗ Start a completely new sketch (or cut and paste).
In this next sketch, we will use the += operator to slow the blink down. This is our starting sketch.

```
int delayPeriod = 1000;


void setup()
{
  pinMode(2, OUTPUT);
}


void loop()
{
  digitalWrite(2, HIGH);
  delay(delayPeriod);
  digitalWrite(2, LOW);
  delay(delayPeriod);
}
```

Notes

If you run this, you get the basic blink sketch.



Sketch A4.6 iterations

Changing the starting delay period to something much shorter, 10 milliseconds, and increasing it by 20 milliseconds on each blink (iteration).

```
int delayPeriod = 10;

void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
  delay(delayPeriod);
  digitalWrite(2, LOW);
  delay(delayPeriod);
  delayPeriod += 20;
}
```

🗒️ Notes

Initially, you see it start off blinking very fast, and gradually the blinks get slower and longer.

🌻 Challenge

Try replacing `delayPeriod += 20;` with `delayPeriod *= 2;`

🛠️ Code Explanation

`delayPeriod += 20;`	We add 10 to the variable delayPeriod on every iteration

Sketch A4.7 increment

If we wanted to stop the blinking at 1,000 milliseconds, we would need to first introduce another variable. We will call this increment and make it bigger, for instance, 50.

```
int delayPeriod = 10;
int increment = 50;

void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
  delay(delayPeriod);
  digitalWrite(2, LOW);
  delay(delayPeriod);
  delayPeriod += increment;
}
```

Notes

We have the same overall effect.

Sketch A4.8 one second delay

Stopping at 1 second (1000 milliseconds), so that when the delay reaches the 1-second mark, it then blinks at that constant rate.

```
int delayPeriod = 10;
int increment = 50;

void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
  delay(delayPeriod);
  digitalWrite(2, LOW);
  delay(delayPeriod);
  delayPeriod += increment;
  if (delayPeriod > 1000)
  {
    increment = 0;
  }
}
```

Notes

It now slows down until it blinks for one second. I timed 10 blinks to make sure.

Challenges

Try the following:

1. Replace `increment = 0;` with `delayPeriod = 0;`
2. Replace `increment = 0;` with `increment = -increment;`