Logic and the Arduino: Bitwise Operators

Binary in code

The Arduino code uses decimal numbers for the source code but also has constant values B0 through B1111111 defined that map binary to decimal values.

The following code needs an Arduino connected but does not require any hardware connections on the Arduino.

```
void setup() {
  Serial.begin(9600);
}
void loop() {
    int a =  B1101;
  Serial.println(a);
  delay(1);
}
```

The Serial Monitor constantly outputs the binary number 1101 as decimal (lucky 13)

Logical operators

The Arduino code can perform logical operations bit by bit on binary numbers.

1 AND 1 = 1

0 AND 0 = 0

1 AND 0 = 0

This operation can be performed on 2 binary numbers comparing each bit in turn

10110 AND

01110 =

00110

Treating the above as their decimal equivalents; 22 AND 14 equals 6. A single & is used for this operation in the Arduino code (&& is the logical AND)

```
void setup() {
  Serial.begin(9600);
}

void loop() {
    int a =  B10110;    // in binary: 22
    int b =  B01110;    // in binary: 14
```

```
    int c = a & b;  // result:   00110 , or 6 in decimal.
   Serial.println(c);
   delay(1);        // delay in between reads for stability
}
```

Viewing the binary in the source code makes the output easier to follow but the regular decimal values can be used instead.

```
void setup() {
  Serial.begin(9600);
}
void loop() {
   int a =  22;
   int b =  14;
   int c = a & b;  // result:   00110 , or 6 in decimal.
   Serial.println(c);
   delay(1);        // delay in between reads for stability
}
```

The bitwise OR is a single pipe |

```
int c = a | b;
```

1 OR 1 = 1

0 OR 0 = 0

1 OR 0 = 1

Working with decimal 22 and 14 we would expect 30

10110 OR

01110 =

11110

The bitwise XOR is a caret ^

```
int c = a ^ b;
```

1 XOR 1 = 0

0 XOR 0 = 0

1 XOR 0 = 1

The XOR of 22 and 14 would be 24

10110 XOR

01110 =

11000

Bitwise NOT is a tilde ~

NOT 1 is 0

NOT 0 is 1

0011 NOT is 1100

```
int c =12;
c = ~c;
Serial.println(c);
```

The code above outputs -13. This is because it sees 12 not as 1100 but as 00001100 and binary numbers beginning with 1 are negative

0000 1100 NOT is 1111 0011