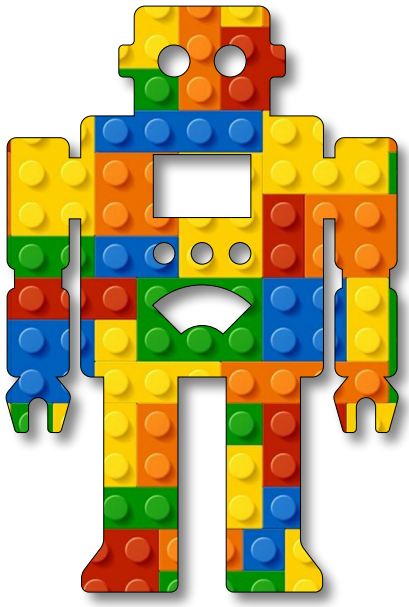
Intelligent Machines Module D Unit #1 truth tables



Content

Module D Unit #1 truth tables for OR and XOR

Introduction to OR and XOR

We will add a second button to our Arduino Nano 33 BLE. It will be attached to digital pin D3. We will use the same built-in LED on pin 13.

We are going to explore a very simple premise. How to use AI to control an LED with inputs from 2 buttons. We will look at two examples. The first...

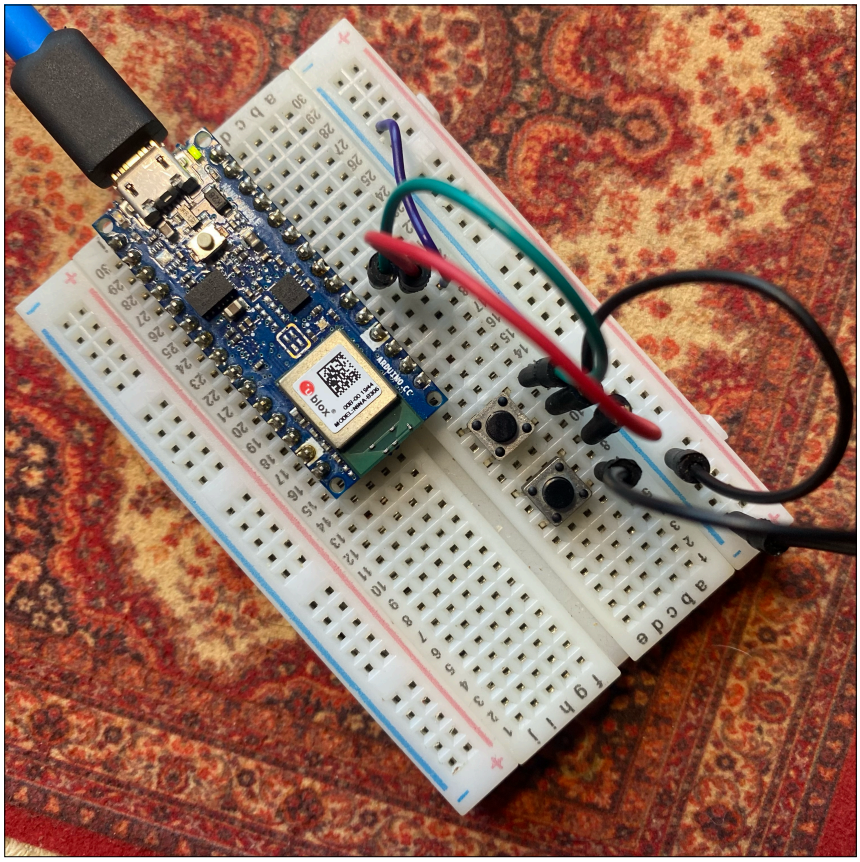
What you will need

The full list is here. Look at the image below and the wiring diagram (fig.5).

1 x Arduino Nano 33 BLE
1 x breadboard
2 x buttons
5 x male-to-male jumper leads

You could add a resistor in series with the button, but the `Arduino Nano 33 BLE` has a built-in resistor that you can pull up. This saves you the bother. The downside is that the logic (`HIGH` = off and `LOW` = on) is reversed.

Figure 1: component setup



Circuit Diagram for the button

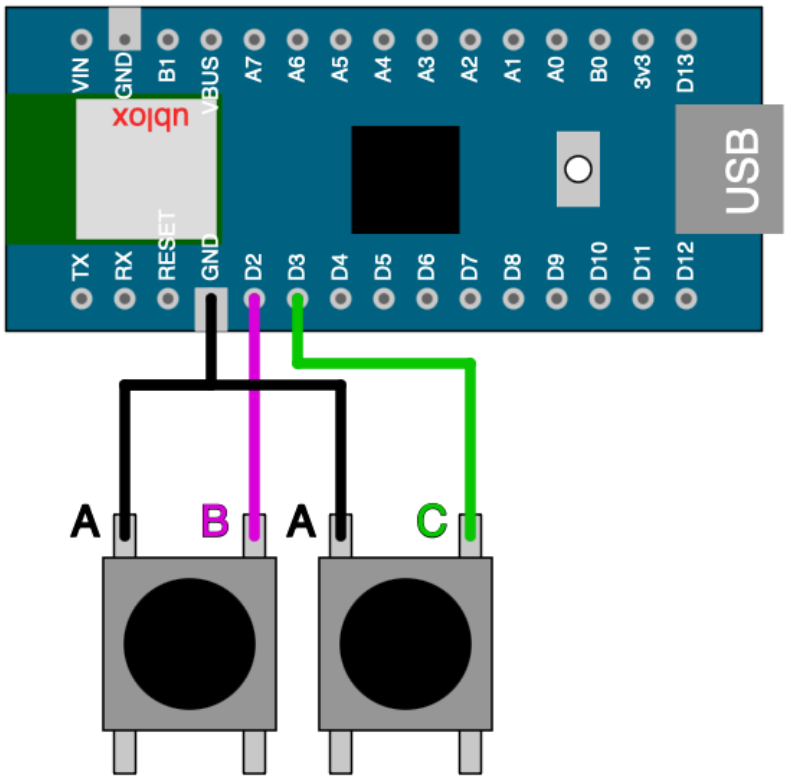
You will need two wires to connect the button to the device.

Button Pins	Arduino Pins
A *(both)*	GND
B	2 *(digital pin D2)*
C	3 *(digital pin D3)*

We are going to add the button to the breadboard. You can put it anywhere away from the board itself.

See Fig.2 below. The wiring diagram shows the connections.

Figure 2: wiring diagram



The OR truth table

A truth table is a way of representing the relationship between inputs and outputs. This helps to clarify what we expect to see. One of the simplest is the OR truth table, which has two inputs and one output. Our two inputs are button 1 and button 2, and our output is the LED, which is either on or off.

We use a simple notion for describing the state of the inputs: 1 or 0. It is quite logical: we give it 1 if the button is pressed and 0 if it is not, and we give the LED 1 if it is on and 0 when it is off. Simple.

OR Truth Table		
Button 1	Button 2	LED
0	0	0
1	0	1
0	1	1
1	1	1

Hopefully, this seems quite logical. If button 1 is pressed, the LED is on; if button 2 is pressed, the LED is also on; if both buttons are pressed, again the LED is on; if no button is pressed, the LED is off. I know I am labouring this point, but a sketch is coming to you soon. We will look at an XOR truth table later.

This is easy to hard-code; we don't really need an AI model to do this, so we will start with hard coding it anyway. We will see if we can use an AI model to do this job for us to illustrate the concept with a simple challenge.

Sketch D1.1 hard coding OR truth table

This is a simple OR configuration using two buttons (inputs) and one LED (output). The LED is on if button 1 or button 2 or both are pressed.

Arduino sketch

```
int ledPin = 13;
int buttonPin1 = 2;
int buttonPin2 = 3;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin1, INPUT_PULLUP);
  pinMode(buttonPin2, INPUT_PULLUP);
}

void loop()
{
  buttonPin1 = digitalRead(2);
  buttonPin2 = digitalRead(3);
  if (buttonPin1 == LOW || buttonPin2 == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```

Notes

Now we have solved the OR truth table with two buttons and an LED, let's look at the XOR problem.

The XOR truth table

This looks almost exactly the same, but there is one significant difference, which I have highlighted in blue. The LED only comes on if only one button is pressed, but not both.

XOR Truth Table		
Button 1	Button 2	LED
0	0	0
1	0	1
0	1	1
1	1	0

Let's go through the logic. If button 1 is pressed, the LED is on; if button 2 is pressed, the LED is also on; but if both buttons are pressed, the LED is off; if no button is pressed, the LED is also off.

This is the XOR truth table, and the difference is significant from a machine learning point of view. The OR truth table is linearly separable and could be solved (probably) by a single perceptron. But the XOR truth table needs a neural network to solve. This is also easy to hard-code, as was the OR example.

Sketch D1.2 hard coding XOR truth table

We just need to change one line of code and add in some more conditional statements.

Arduino sketch

```
int ledPin = 13;
int buttonPin1 = 2;
int buttonPin2 = 3;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin1, INPUT_PULLUP);
  pinMode(buttonPin2, INPUT_PULLUP);
}

void loop()
{
  buttonPin1 = digitalRead(2);
  buttonPin2 = digitalRead(3);
  if ((buttonPin1 == LOW || buttonPin2 == LOW) && (!(buttonPin1 == LOW && buttonPin2 == LOW)))
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```

🗒️ Notes

This should work according to the truth table for the XOR problem.

🛠️ Code Explanation

`&& (!(buttonPin1 == LOW && buttonPin2 == LOW)))`	Returns true as long as button 1 and button 2 are not pressed at the same time