ENGDUINO LIBRARIES: THE BUTTON

There is a single user button on the Engduino and we can work out when it is pressed (or released) to help us control what is happening in the program.

HEADER

At the top of your program you must have the following line – often in addition to the one for LEDs and other bits of the Engduino.

```
#include <EngduinoButton.h>
```

SETUP()

At the top of your program you must have the following line – often in addition to the one for LEDs and other bits of the Engduino.

```
EngduinoButton.begin();
```

BASIC FUNCTIONS

The first functions allow you to wait until a button is pressed or released....

```
EngduinoButton.waitUntilPressed();
```

or

```
EngduinoButton.waitUntilReleased();
```

If you have these in your program, then it will do nothing until the button is pressed (or released). For example the following code turns your Engduino into a torch – press the button to switch it on and off:

```
...

void loop() {

      EngduinoButton.waitUntilPressed();
      EngduinoLEDs.setAll(WHITE);
      EngduinoButton.waitUntilPressed();
      EngduinoLEDs.setAll(OFF);

}
```

The second function needs you to know a little bit more about programming. In a program, there is a way in which we can choose to do one thing or another – it's called the 'if' statement. It's easiest to see by example with our next function.

```
void loop() {

      if (EngduinoButton.isPressed())
            EngduinoLEDs.setAll(RED);
      else
            EngduinoLEDs.setAll(OFF);

}
```

This switches the LEDs to red only when the button is pressed. As soon as you release the button, then it sets the LEDs off.

MORE ADVANCED FUNCTIONS

The final function is useful is you want to write code that occasionally checks whether the button has been pushed but then goes on to do something else. While the code is doing the something else, it might be the case that the user pushes the button and lets it go and, because you haven't checked to see if it's pressed then, you might miss it. Well, the following function will tell you whether a button has been pushed or released since you last called that function - while your code is doing something else. In this way you need never miss a button press again.

```
void loop() {

      if (EngduinoButton.wasPressed())
            EngduinoLEDs.setAll(BLUE);
      else
            EngduinoLEDs.setAll(OFF);

      delay(1000);

}
```

What you'll see is that the LEDs turn blue for a second if you press the button, and then they go off again. If you press the button while the program is delaying, then the LEDs don't come on immediately, but they will come on when it next tests to see if the button has been pressed. You'll see this more clearly if you make the delay bigger.

Now try using **`EngduinoButton.isPressed()`**instead of **`EngduinoButton.wasPressed()`**. What you will find is that if you press the button while the code is delaying the Engduino will never realised that you pressed it at all.

IDEAS FOR PROJECTS WITH THE BUTTON

- Traffic lights that change when you press the button
- Games that need user input…