The Engduino

At heart, the Engduino is a simple, inexpensive, computer, designed primarily for teaching programming in a way that is interesting.

The Engduino’s hardware is like that of the Arduino LilyPad and it inherits many of the great features of the Arduino – for example, we use the same simple programming environment, which can run on PCs, Linux, and MACs. Like other Arduinos, it can’t be programmed in Python – the Arduinos have a special language that is more like C++. Because the Arduino is so popular, there are many good books about this.

There are many derivatives of the Arduino and other similar systems; however, the Engduino comes with some added benefits:

<table>
<thead>
<tr>
<th>It has a battery – which means you can use it wherever you want, not just when it’s connected to a mains lead or another computer.</th>
</tr>
</thead>
</table>

| It has 16 multicolour LEDs that can be lit with red, green, and blue, and any mixture of these. For example, one might mix these primary colours to get the secondary colours... |

| Actually, the colour of the LEDs can be set to even more colours than the primary and secondary colours – and the brightness can be set as well. In total, there are 4096 different possibilities with different mixtures of red, green and blue. |
There is a button – which can be used for whatever you would like – to make a choice between different options, to play a game, .... it’s up to you and your program.

The Engduino can measure temperature electronically, and give you a number in degrees Celsius

The Engduino can measure acceleration using a device called an accelerometer. According to Newton, force and acceleration are related to each other... \( F = ma \) (force = mass x acceleration)

The acceleration that the Engduino measures could be thought of as being composed of two parts:

- Dynamic acceleration – which you get when a car accelerates, for example. Or when a human moves – any change in direction is an acceleration, so moving up and down when walking involves acceleration, and we can measure that (this is how a pedometer works).
- Acceleration (or force) – due to gravity. Even when the Engduino is on a desk, not apparently moving, it will register the force due to gravity. Since the accelerometer can tell you in which direction the force is acting, we can use this to turn the Engduino into a spirit level.

Any action that involves a change in direction – any sufficiently large vibration – can be measured with an accelerometer. Likewise, we can know whether the Engduino is face up or face down, or being turned end over end.
The Engduino can communicate with each other – over a short range of maybe a metre or so – using an infra-red beam. This works in much the same way as the remote control on your TV, but the signal is weaker and so it won’t travel so far.

This means that the Engduino can pass messages between each other. Say we wanted to make a game like Simon ([http://en.wikipedia.org/wiki/Simon_%28game%29](http://en.wikipedia.org/wiki/Simon_%28game%29)) in which two players had to recreate a pattern of alternating flashes on their Engduino by pushing the buttons in the right order. We need the Engduino to communicate to decide on the order in which to flash the LEDs, and to work out whether the buttons were pressed in the right order.

Communications is fundamental to the Internet, and the IR system on the Engduino allows people to start to understand how communications works.

Like Arduinos, the Engduino can communicate with a computer over the USB – so you see what it is doing and you can communicate with it from, say, a Python or Java program simply by reading from and writing to the serial port. This means you can write your own GUI, allowing, say, the LEDs to be set, or the temperature to be displayed – and you can run this program on a PC, a Raspberry Pi or whatever you please.

Finally, again like Arduinos, you can interface the Engduino to other sensors. For example, we have interfaced ours to a pulse sensor ([http://pulsesensor.com/](http://pulsesensor.com/)) so the LEDs flash along with your heartbeat.

This is just the start. The Engduino is really a development board – it has the potential to to be turned into lots of things, either using the LEDs and sensors we have built in, or using other devices you might add. It can work with PCs, MACs, the Raspberry Pi and other little computers.

The main remaining ingredient is creativity. That’s where you come in....
**Built-in demos (CombinedDemos in the examples folder)**

When the demo starts up (or after you've pressed the reset button), there is a running light on the LHS of the Engduino and one pushes the button a number of times both to light LEDs on the RHS of the Engduino and to select the appropriate demo. The demo is selected after the button has not been pressed for 3 seconds. Once selected, the chosen demo will run until the device is reset.

Demos:

1. RunningLEDs
2. Level
3. TemperatureLEDs
4. Comms
5. FizzBuzz
6. ReactionGame
7. External interfacing - PulseSensor
8. Serial communication
9. MoodLighting

In more detail:

1. **LEDs demo**
   The aim of this example is to show some of the methods of causing the LEDs to light. The first part of the sketch is where red, green and blue LEDs chase anti-clockwise around the Engduino, getting gradually faster until they blur into white. When going sufficiently fast, we display red green and blue on all LEDs, followed by a range of colours.

2. **Accelerometer demo – level**
   Show a red LED on one side if we're low on that side show nothing if we're high, and show green on both sides if it's level(ish). We delay between readings because it's a bit flashy otherwise - too hard to hold steady.

3. **Temperature LEDs**
   In this sketch, the colour of the LEDs changes on a linear scale from blue (cold) to red (hot) via mixtures of the two colours in proportion to the temperature. The temperatures that define blue and red are given in the #defines below. Temperatures outside the range, are capped so those lower than TBLUE are rendered as blue and those higher than TRED are red. The sketch illustrates the use of both thermistor and LEDs.

4. **Send message**
   This tests the IR communications. Pressing a button on an Engduino causes a message to be sent. The LEDs flash blue when the message (“Hello World”) is sent. An Engduino receiving the message will flash green if it receives the message correctly, and red if not. Illustrates use of IR, LEDs and the button

5. **FizzBuzz**
   This is a game like fizz-buzz where you have to do mental arithmetic, keeping a running total of the count of LEDs and pushing the button when that count is a multiple of three or five
6. Reaction Game
This is a game where you must push the button within a short time of seeing a blue LED. If you push the button without a blue LED or if you push too late, you lose and you start again. If you succeed, things get faster.

7. External interfacing
Not really useful to you...
This is a sophisticated example of interfacing to the Engduino. It makes use of a pulse sensor that can be obtained from [www.pulsesensor.com](http://www.pulsesensor.com). This returns an analogue signal, which is picked up on the IO_1 expansion pin and the code does some processing to determine which part of a heartbeat the signal belongs to. It outputs this in the form of flashing LEDs and, in more detail over the serial link. This can be shown on a GUI, an example of which is included as 'Processing' code. To run that code you will need to download Processing from [http://processing.org/](http://processing.org/). Clearly it would also be possible to write a similar interface in Python, Java, or the web technology of your choice.

8. Serial communication
This is a very simple program to demonstrate that it is possible to communicate with the Engduino over the serial connection - both from and to the Engduino. One example use for this is to communicate with a program written in Python on a Raspberry Pi - the bulk of the code can be written on the Pi and the Engduino used for illuminating LEDs or as a sensor input to the Pi.

9. Mood lighting
The LEDs gradually change colour from one random colour to another, in an amount of time that is random but bounded. This demonstrates a slowly moving change in the rgb value given to the LEDs using an accumulator which ensures that each channel changes at an even rate given a time period in which to change.
The code for this is in the CombinedDemos folder of the examples directory. The demos are also available as stand-alone files.