ENGDUINO LIBRARIES: THE INFRARED

The infrared transceiver allows two (or more) Engduinos to talk to each other over a distance of upto about 50-100cm. This is the most complex of the libraries in terms of basic functionality, but it is easy enough to use once you are a competent programmer.

HEADER

```
#include <EngduinoIR.h>
```

SETUP()

```
EngduinoIR.begin();
```

BASIC FUNCTIONS

To make use of the IR, you must understand the concept of arrays. Take some time to revise this if you need to before trying the exercises in here. Because the IR is about communication, you will need two Engduinos to do anything useful.

In terms of basic functions, you can send a single character (byte) at a time. So, for example, to send the character 'c' we do the following.

```
EngduinoIR.send('c');
```

And, to receive, you should execute the following code on the destination:

```
uint8_t buf[IRBUFSZ];             // Declare a buffer in which to receive
                                  // any message. It should be this size.

int len = EngduinoIR.recv(buf);   // Block until there is something to
                                  // receive and then receive it.
                                  // len contains the length of the
                                  // received message
```

Write code for a sender, which sends a character every second. On a different Engduino, write the code for a receiver, which repeatedly executes a receive and prints the result.

MORE ADVANCED FUNCTIONS

Rather than sending a single byte at a time, you can send upto 12 bytes:

```
char buf[12];                     // Declare a buffer of characters.
                                  // [Could be declared as uint8_t]

... put some values into buf

EngduinoIR.send(buf, 12);         // Send the buffer - you need to provide
                                  // the length of the buffer as well as
                                  // the contents
```

If you don't want to wait forever for an incoming message to arrive, you can set a timeout. When the function call returns, either (a) a message has been received or (b) the call has timed out. In the first case, the length of the message is returned, in the second, a negative number is returned:

```
uint8_t buf[IRBUFSZ];             // Declare a buffer in which to receive
                                  // any message. It should be this size.

int len = EngduinoIR.recv(buf, 500);   // Block until either we have a
                                  // message to receive, or 500
                                  // microseconds has elapsed
                                  // len contains either the length of the
                                  // received message or a negative number
if (len >= 0)
    .... a message was received
else
    .... the call timed out - there is no message
```

If you set the timeout to zero, the **recv** function will block forever - as it did when we omitted the timeout value.

IDEAS FOR PROJECTS WITH THE IR

- Make an Engduino send the word “Hello” when a button is pressed, and print out any message it receives. Try this between two Engduinos and vary the distance between them. You will probably notice that at some point, the message isn’t received correctly.
- To deal with this, we need to know when a message has been received correctly. There are several ways of doing this: for example, you could use a parity bit or a cyclic redundancy check (CRC) algorithm. Search online for these terms and see whether you can implement them for the Engduino
- Write a simple communications protocol. Each Engduino should periodically send a message that tells other Engduinos that it is there. It should also listen for other Engduinos sending this message. When two Engduinos meet, one of them should flash the LEDs red, the other should flash them blue. It doesn’t matter which does what, so long as one is red and the other blue. You shouldn’t use the button at all – this should happen automatically when they are in range of each other.