PI-LITE Users' GUIDE

In this guide we will:

1.Tell you what the Pi-Lite is and what it can do;

2.Show you how to set up the Raspberry Pi to work with the Pi-Lite;

3.Show what you can do with the Pi-Lite and its pre-loaded software;

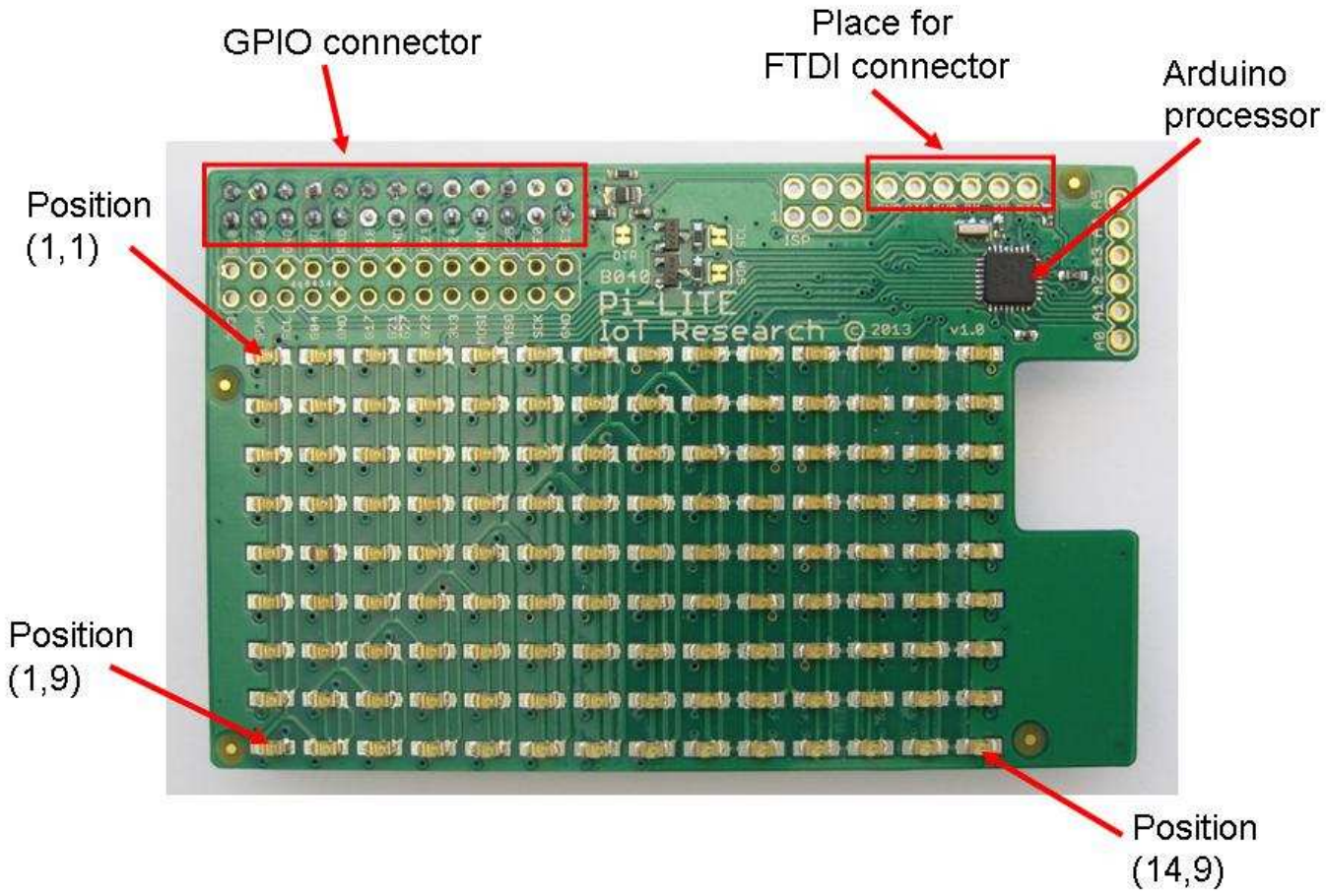
4.Point you at the examples that we have prepared for you, so you get going sooner and faster;

5.Show how to drive the Pi-Lite without the Raspberry Pi;

6.Explain how to program the Arduino processor on the Pi-Lite from any computer, even the Raspberry Pi itself.

Introducing the Pi-Lite

The Pi-Lite is a very versatile 9 x 14 LED matrix display with an on-board Arduino ATmega 328 processor. Each pixel is individually addressable, so you can display anything in the grid. The Pi-Lite is a fully assembled board. It was primarilly designed as a Raspberry Pi add-on board, but works equally well without the Raspberry Pi. The idea comes from the very popular Arduino LoL shield by Jimmy Rodgers and brings the capabilities of such a shield to the world of the Raspberry Pi. LoL stands for "Lots of LEDs" of course!!



You drive the Pi-Lite via its serial port:

1.it is really simple to send text and graphics to the 126 LEDs from anything that addresses the serial port, such a program, or a serial monitor.
2.you can program the on-board Arduino processor with your own program, replacing the pre-loaded Ciseco software

You can access the serial port in several different ways:

1.By plugging the Pi-Lite into the GPIO pins of your Raspberry Pi you access it via the Raspberry Pi's serial comms port at 9600bps.

2. By using the serial port brought out on the FTDI connector, you can use other devices than the Raspberry Pi, such as a PC, a Mac or Linux machine.

The LED matrix is controlled by an ATmega 328 processor, which is pre-loaded with Ciseco software. This pre-loaded software is based on, and improves, the original Jimmy Rogers version. The Pi-Lite will run any Jimmy Rogers or similar sketch. It will also run your own sketch if you want to write one.

With a suitable stand-alone programme, the Pi-Lite can run on its own, without a connection to a serial port. For instance, you can run your display or scrolling message on battery or mains.

If you want to get a flavour of what the Pi-Lite with its pre-loaded software is like before buying one, we have an emulator for you to play with. Not as good as the real thing, but it should whet your appetite.

Setting up the Raspberry Pi for the Pi-Lite basic functions

If you use the Pi-Lite with your Raspberry Pi, please make sure you follow the instructions in this guide before you plug in the Pi-Lite. It will ensure your Raspberry Pi is set up correctly to work with the Pi-Lite (and incidentally, with other Ciseco products too).

If you don't want to go through this set up, you can purchase the pre-configured Ciseco SD card which has the Raspberry Pi Wheezy image with the changes needed for Ciseco hardware to work correctly. You can also download the Ciseco Wheezy image and make your own SD card: that way you don't need to spend money!

Using the Pi-Lite pre-loaded software

This section refers to the Ciseco software that ships with the Pi-Lite.

Once the Pi-Lite is powered up and running you can communicate with it via the serial port. The serial port can be accessed in two different ways:

1. From a serial terminal or program on the Raspberry Pi when plugged into the GPIO
2. From the FTDI connector (described in a following section below).

When plugged into the Raspberry Pi, the serial port is likely to show up as /dev/ttyAMA0, but on your specific machine or controller it may be a different port. You can use minicom or another similar terminal program to access it:

minicom -b 9600 -o -D /dev/ttyAMA0

Of course you can also address the serial port from a program you have written on the Pi. See the section on examples for inspiration.

Note: you can find the preloaded software in sketchbook>libraries>LoLShield>examples>PI_LITE

General rules

The general rules are as follows:

1.You can send any text, which will be scrolled on the Pi-Lite.
2.When $$$ is received, all scrolling stops and the Pi-Lite enters command mode. See the command reference below. "$$$" will not be output on the scrolling display.
3.Scrolling will start again when a non-command character is received.
4.All control characters, including line feed (<LF>) are ignored at all times. The only exception is carriage return (<CR>) which terminates a command, but is otherwise ignored.
5.Any data received other than commands switches the display to scroll mode and displays the character via scrolling.

Power up sequence

On powerup the Pi-Lite will send identification and version down the serial port

"Pi-Lite version n.n"

Note, this will not show on the LEDs.

The following sequence is then run through:

1.delay for 1 second (0.5 for the Uno bootoloader and an additional 0.5 waiting for $$$ if you wish to silently interupt to avoid the splash screen)
2.scroll a single row down and up
3.scroll a single column right and left
4.Display Pi-Lite Logo with firmware version number (denoted by single LED's top right)

This sequence can be interrupted by sending a $$$.

Display lay out

Pixel (1,1) is upper left and (14,9) is lower right.

<needs a picture to show top and bottom for those who may be confused about it>

GPIO connector is at the top left hand side.

Command mode

Entering command mode:

When $$$ is received the Pi-Lite responds with "OK" and enters command mode.

All commands must be terminated by a carriage return

<command><CR>

If <command><CR> is not received within a few seconds of $$$, the command terminated and ignored.

After receiving and carrying out a command, the Pi-Lite leaves command mode, and returns to scroll mode.

If invalid data is received whilst expecting a command character then the command is ignored and command mode terminated.

SPEED(value) $$$SPEED200[RETURN]	**SPEED** Set scrolling delay in mS 1 is scrolling very fast, 1000 is scrolling very slow. Default speed is 80.
F(value) example (pacman) $$$F000000000000000000011100001111111001111111011111111111111011111111011000110011000110000000000000000000000000000000	**Frame buffer** (one digit per pixel - 126 digits in total) (value) is a 126 character string in one's and zeros to represent each LED state (on or off)
B(column),(value) examples $$$B1,89[RETURN] - column 1 set to 89% $$$B14,20[RETURN] - column 14 set to 20%	**Bargraph** (column) 1 to 14 (value) 0 to 100% NB these commands do not over write the screen, this is by design to allow mixed modes, if you want a clear screen, issue an ALL,OFF
V(row),(value) examples $$$V1,89[RETURN] - bar 1 set to 89% $$$V2,20[RETURN] - bar 2 set to 20%	**VU meter** (horizontal 2 rows) set row r (1-2) to val in (row) 1 or 2 (value) 1 to 100% NB these commands do not over write the screen, this is by design to allow mixed modes, if you want a clear screen, issue an ALL,OFF
P(column),(row),(action) examples $$$P4,4,ON[RETURN] $$$P4,4,OFF[RETURN] $$$P4,4,TOGGLE[RETURN]	**Pixel** set a pixel state (column) 1 to 14 (row) 1 to 9 (action) ON,OFF,TOGGLE
ALL,(value) examples $$$ALL,ON $$$ALL,OFF	**ALL** set all the pixels on or off (value) ON or OFF
SCROLL(value) examples $$$SCROLL-4[RETURN]	**Scroll** val columns left or right (value) -1 to -14 will shift to the right (value) 1 to 14 will shift to the left Once a pixel is off the display, it won't come back when you scroll back
T(column),(row),(character) examples $$$T1,1,Z[RETURN] - display a Z at location 4,4	**Text** display at location x,y a text character

Examples

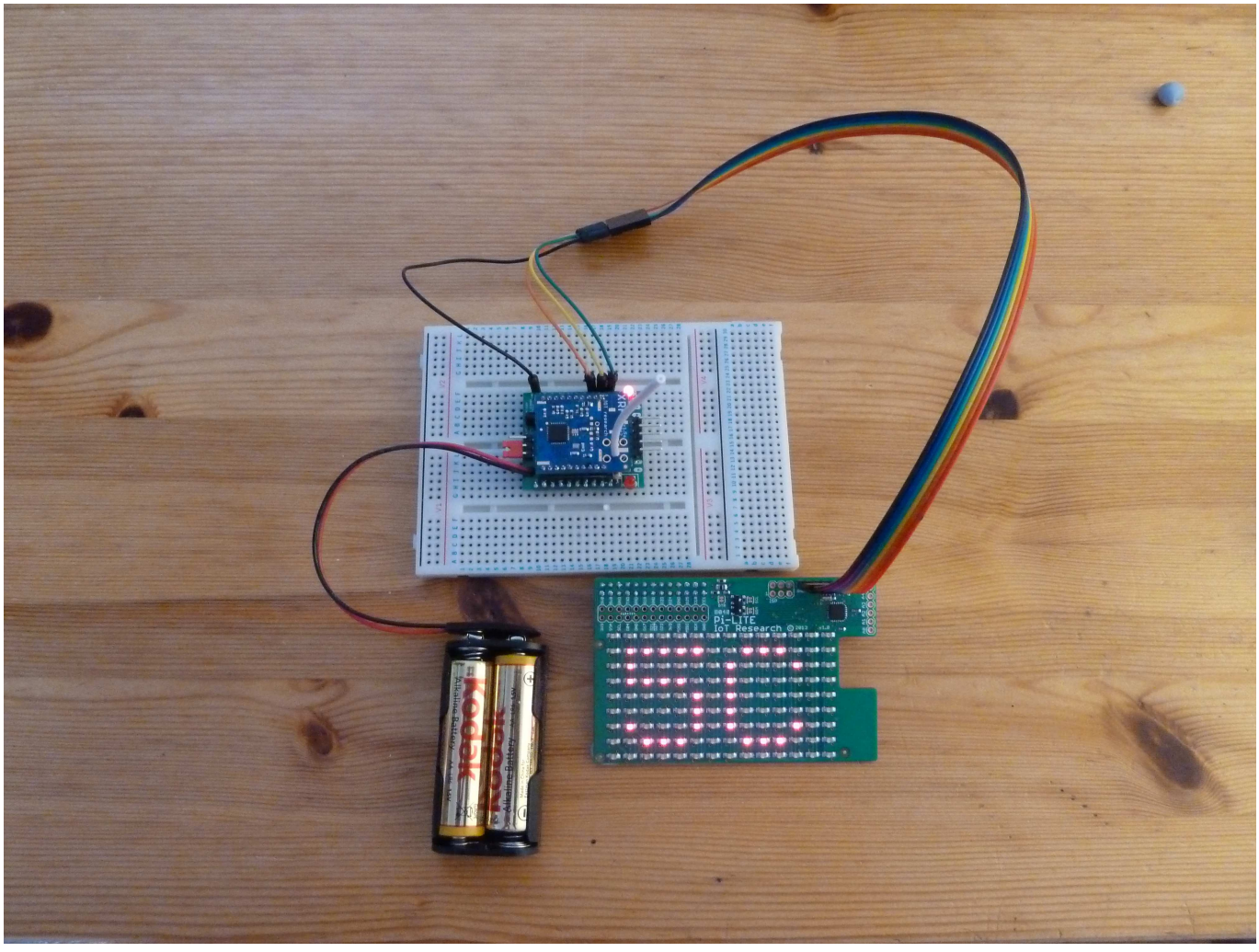
We have supplied a range of examples to show you some of the things you can do with the Pi-Lite. You can use these examples to create your own solutions, as the code is open source. The examples are described in a series of pages, starting here.

Running the Pi-Lite without a Raspberry Pi

The Pi-Lite has an on-board Arduino with pre-loaded software. By supplying power to the Pi-Lite board, you can use either the serial port (GPIO pin 10, UART-RXD and GPIO pin 8, UART-TXD), or the FTDI pins to send data and commands to drive the display. You need to be careful to ensure you use 3.3V or 5V serial signalling.

Example: a wireless display unit

As an example, the picture below shows the Pi-Lite (with its default firmware) connected to an XRF, after we sent it the string 5C$$$ from a PC. This scrolls in the text 5C and then halts scrolling due to the command initiator: $$$.



We connected the Pi-Lite up via the FTDI connector pins to an XRF Breakout Board (XBBO) with an XRF on board. This makes it easy to place the whole system on a breadboard. Here is how we connected the two together:

	Pi-Lite FTDI pin	XRF pin
	GND	10 (GND)
	CTS	n/c
	5V0	1 (VDD)
	RX	2 (DOUT)
	TX	3 (DIN)
	DTR	n/c

We then used an SRF stick connected to a PC (not shown in the picture) to wirelessly send data and commands to the Pi-Lite for display. You can of course send it data from a program too. Why not try and make it display messages from your Twitter feed for instance?

Note that the Pi-Lite will work with the 3V supplied. You will probably want to use a larger capacity battery than shown.

Programming the Pi-Lite

If you want to run anything other than the pre-configured program on the Arduino on the Pi-Lite board, you can of course do so. The Pi-Lite can be programmed from any computer with the Arduino IDE installed on it.

Programming from anything but the Raspberry Pi

If you use a computer other than the Raspberry Pi, there is nothing you need to do, other than corectly connect the Pi-Lite to the serial port of the computer you are using. Simply connect the Pi-Lite to your computer in one of two ways:

1.Via the FTDI pins on the Pi_Lite. You may like to solder a socket or some pins into the Pi-Lite board to do this.

2.Via the serial port, provided on the Pi-Lite board. Again adding a socket or pins makes life easier.

You can then use your Arduino IDE as you do for any other Arduino based board.

Programming from the Raspberry Pi

If you want to use your Raspberry Pi to program the Arduino on the Pi-Lite board via the GPIO, you will need to set the Raspberry Pi up correctly before you can do so. In particular, you need to do the following:

1. Install the Arduino IDE with some special wrappers
2. Install the Pi-Lite libraries
3. Depending on which libraries you installed, you must apply a fix to make sure the Raspberry Pi continues to boot and halt correctly.

We will take you through each step below.

Step 1: Installing the Arduino IDE with wrappers on the Raspberry Pi

Please carefully folow the instructions in this guide to install the Arduino IDE with its wrappers and ensure they are set up correctly for use via the GPIO.

You **must** follow these steps for things to work correctly and before you progress with step 2 below.

Step 2: Installing the libraries

Next you need to install the LoL library for the Pi-Lite.

Although the Pi-Lite will work very well with the Jimmy Rodger's LoL libraries, we recommend that you install the Ciseco upgraded LoL library. This is in part because the compiler on the Raspberry Pi is more fussy than the older compilers in use when Jimmy's library was released. The Ciseco library

1. Includes the default program we installed on the Pi-Lite before shipping
2. Include a number of enhancements and improvements that make the Pi-Lite do things LoL cannot;
3. Includes a fix that ensures your Pi will continue to boot correctly once you have uploaded your own program.

So, if you use the Ciseco LoL library, you do not have to carry out step 4 below!

The Ciseco LoL library is available form the github here: https://github.com/CisecoPlc/PiLite. First download the library zip file. To do so press the **ZIP** icon and save the file. You should see a file called **PiLite-master.zip** appear in .

If you have not been seduced in running the Arduino IDE for the first time, then do it now. This will create a folder called **sketchbook** in your home directory. We need this directory to install the libraries.

Close the Arduino IDE for now.

Then take a look in your sketchbook folder, likely to be in **/home/pi/sketchbook**. If there is a folder called **libraries** in the sketchbook, all well and good. If there is no such folder create one.

Double-click the file **PiLite-master.zip** you downloaded earlier. This opens the Xarchiver.

Select Action>Extract and in then click Extract, to extract the **PiLite-master** folder.

Open the **PiLite-master** folder and you will see a folder called **LoLShield**. You need to copy this folder to your libraries folder in **/home/pi/sketchbook/libraries**.

That should be enough to "install" the library.

Now start the Arduino IDE and you should be able to see a set of LoL examples appear in the examples drop down of the Arduino IDE. If this is so, you have successfully installed the library.

If you have chosen to use the Ciseco LoL library, you are done. Try some of examples and use them to inspire you to make your own program do exactly as you want.

If you have chosen to use the Jimmy Rodgers LoL library, you **must** continue and carry out **step 4** below to make sure everything keeps working without problems.

BTW, there is some information on the Arduino site on how to install libraries if you are interested.

Step 3: LoL libary fix to avoid boot and halt problems

If you elect to use Jimmy Rodger's LoL libraries on the Raspberry Pi, then you need to make a small change to a file to ensure your Raspberry Pi booting and halting works correctly when the Pi-Lite with your Arduino sketch on it, is connected. Here is what to do:

Find the file **Charlieplexing.cpp** in the folder **/home/pi/sketchbook/libraries/LoLShield**

Go to line 104, which will look like this:

```
#define MEASURE_ISR_TIME
```

comment this line out by adding two slashes to the front of it. It should look like this:

```
//#define MEASURE_ISR_TIME
```

Save the file and exit and, that's it.

For those of you curious about the why and wherefores of this step, here is a brief explanation: When the Raspberry Pi boots up it looks to see if pin 5 on the GPIO is low. If it is, it starts in safe mode, instead of booting normally. Alternatively, when the Raspberry Pi is halted, it can be woken up by bringing pin 5 down. The Pi-Lite has a hardware option to work in SCI mode and uses pin 5 for SCL. This can interfere with booting or halting of the Raspberry Pi, so the change to **Charlieplexing.cpp** ensures that this won't happen.

What next?

First, have fun experimenting! For instance, you could try out the Twitter feed for the Pi-Lite or one of the many other examples and get some inspiration for

your own project.

Second, please share your experience and your projects on our Forum.