Getting Started Guide

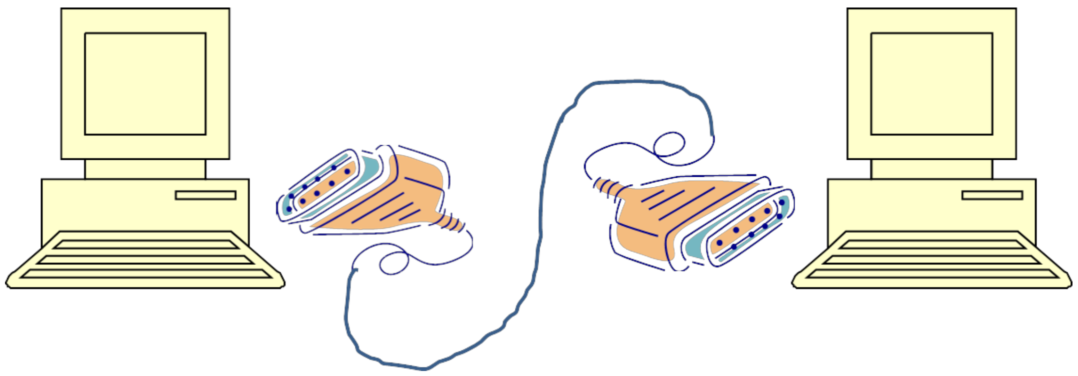
Having read our product introduction (http://shop.ciseco.co.uk/ciseco-wireless-introduction/), you probably have an idea about what you need to buy for your project. But you may well be worried about how things will actually fit together when you receive your kit.

This document is designed to show that things are perhaps a lot easier than they may at first appear. We start by using Ciseco radio units in serial links between computers and microcontrollers. We then take a look at how communication with sensors is achieved.

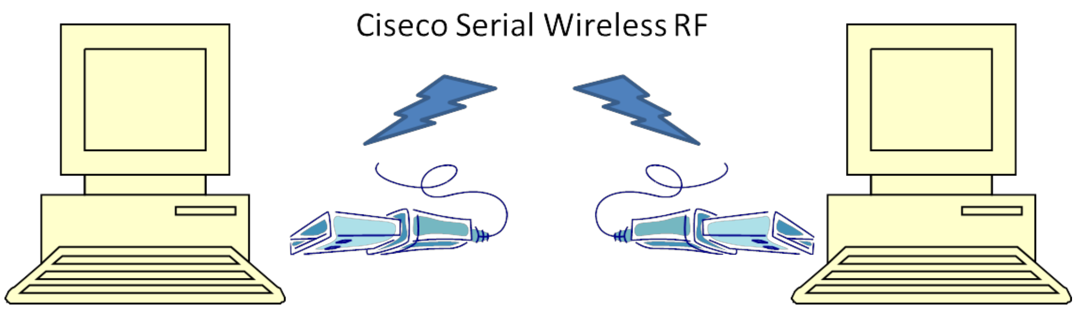
Before getting into the nitty gritty, I have some great news: the Wireless Inventors Kit for the Raspberry Pi (RasWIK) allows *anyone* with a Raspberry Pi to build a wireless project in minutes. People say it is the easiest introduction into wireless control systems they have ever seen. So, you may well want to check it out, as well as read the rest of this document.

Connecting two computers

Remember how we used to connect computers together before there was Ethernet and Wi-Fi everywhere? Yes, we did it using serial RS232 connections:



Our radio modules basically turn a serial cable connection into a wireless serial connection:



You may remember that serial connections can operate at a variety of different speeds (baud rates), with or withour start and stop bits etc. We used to use modem commands on each side to configure the connection before sending and receiving data. This was done with AT commands or modem commands. Our wireless radio connections can also be configured using AT commands, but everything we describe in this document can be achieved with the default parameters. All you need to remember now is that AT commands are connection configuration commands, not data transmit/receive commands.

Most modern computers no longer have a serial port, but they do have Universal Serial Bus (USB) ports. Using the Future Technology Devices International (FTDI) specification and chip, it is common to turn a USB port into a virtual serial RS232, COM or terminal port.

Ciseco SRF stick and URF units use an FTDI chip between the serial Receive and Transmit (RX/TX) pins presented by the radio and the USB connector. So, when you plug the SRF stick or URF into your computer, an appropriate driver will present the operating system with a serial port. On Windows it is necessary to install a driver for this to work. On Linux (or Mac) you do not need to install a driver.

Please refer to this guide to set up your PC to work with Ciseco hardware.

If you have a Linux computer or a Mac, you simply plug in the SRF stick or URF into one of the USB ports and then search for the added device in /dev. On many Ubuntu machines this comes up as /dev/ttyACM0, on the Raspberry Pi look for /dev/ttyAMA0.

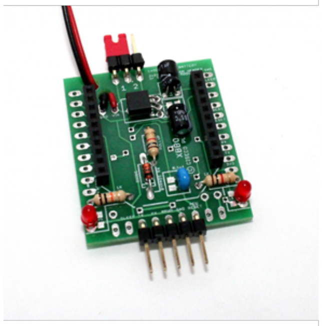
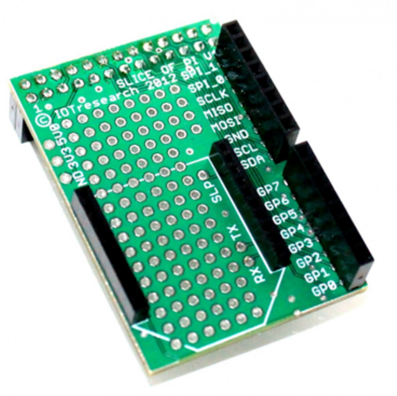
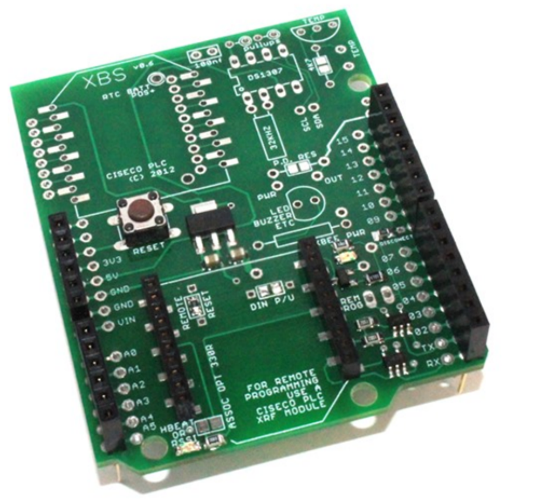
On Linux (including the Raspberry Pi) you have even more choice for terminal emulator programs and if you are a regular Linux user, you will no doubt have your favourite program.

Connecting to your microcontroller

Most small microcontrollers do not have Ethernet or Wi-Fi like regular computers. The software required to run the required protocols could overwhelm such small controllers or require them to run with large memories and bigger power supplies / batteries than you would want. Instead, our radios simply present the microcontroller with a standard serial port through which to communicate. They do not know if this is wired or wireless; it is just a serial data port to them.

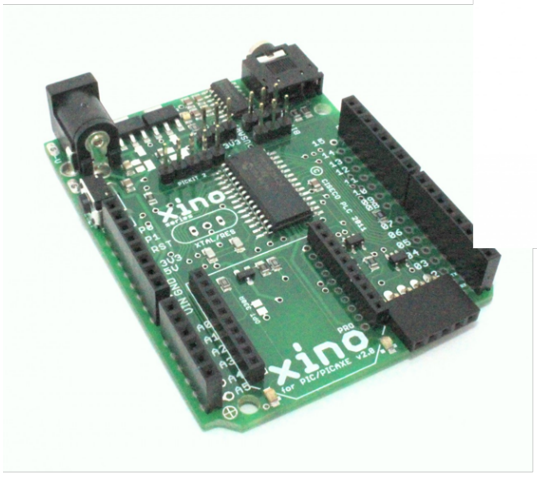
The first thing to do is to connect one of our radio units to the microcontroller you have. We have several options for standard microcontrollers:

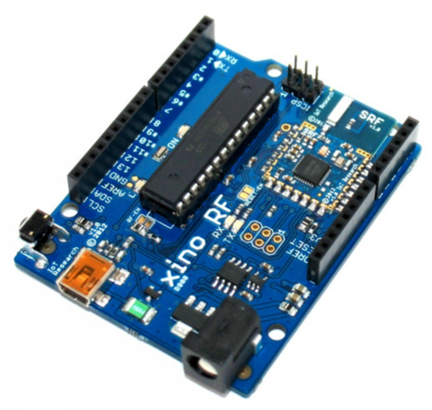
Interface board	Connects to	Radio unit
Active Xbee breakout board	Arduino basic, PIC, PICAXE,	XRF
Slice of Pi	Raspberry Pi	XRF
Sliced of Radio	Raspberry Pi	SRF
XBee Shield	Arduino UNO	XRF

Alternatively, you may have bought a microcontroller with an on-board socket for one of our radio units or one with a surface mounted radio (SRF) included:

Microcontroller board	Radio unit
Xino Pro for PIC	XRF
Xino Pro for PICAXE	XRF
RFμ-328 - Arduino ATMega 328	SRF
XinoRF - 100% Arduino UNO R3	SRF





To communicate with microcontrollers, you need to run software on them so they can listen to and/or send data on the serial port. You can write a sketch (Arduino) or a program on your

Raspberry Pi to interact with the serial port and do things with the I/O pins on your controller. The possibilities are of course endless and limited only by your enthusiasm and ability.

Note that on the Raspberry Pi: You need to set your Pi up to work with Ciseco hardware. This set-up guide shows you how easy that is.

Communicating with your microcontroller

Ok, so now we have all the bits together to communicate from a regular computer, via one of its USB ports and a URF to an XRF or SRF on a remote micro-controller. What shall we communicate?

To help you get started, we have developed a very simple protocol called Pinata and written a few sample programs to execute the protocol. Pinata is a very simple lightweight protocol that allows individual pin control of many popular micro controllers without any programming knowledge. Each serial transmission is 12 characters that are human readable and allows the status of a single pin to be controlled or read. Depending on the capabilities of the micro, pins may be set high, low, or have a PWM value applied, or read as a digital or analog input.

It’s ideal for workshops or educational environments where the focus is on making things happen rather than learning to write code first. It also suits the more skilled developer that may wish to quickly and easily test a prototype or build a mock-up.

Pinata software is currently available for the following micros;

- Atmel ATMega328 (Arduino and Arduino clones)
- PICAXE
- mbed (LPC1768 & LPC11U24)
- Raspberry Pi

For more details: http://openmicros.org/index.php/articles/93-llap-lightweight-local-automation-protocol/pinata-the-easy-way-to-rf-control-your-micro/200-pinata

Sensors and actuators

Remember that the Internet of Things is all about sensors and actuators? So far, all we have talked about is connecting computers to one another and with microcontrollers. In this section I shall explain how sensors and actuators fit in and how we see that evolving.

Our TI 111x based radio units (XRF, ERF, URF) have three main on-chip functions

1. Radio transmitter and receiver (transceiver)
2. Micro-processor
3. Encryption hardware

The micro-processor has three functions:

1. First it runs the software that controls the radio transceiver and the encryption hardware.
2. It runs the boot loader that allows programs to be downloaded to it.
3. It can also act as a microcontroller in its own right and runs so-called “personalities” that control its pins, take measurements etc.

Out of the box, the radio units ship with radio control and boot loader functions and serial pass-through personality (firmware). This allows you to easily build the serial connections described in the previous sections.

We provide a range of sensor and actuator personalities. Each is encapsulated in firmware that needs to be written to the radio device (typically an XRF) to give your radio unit a specific predefined function.

The firmware for these personalities is available on github (https://github.com/CisecoPlc/XRF-Firmware-downloads/downloads). Note these are the latest versions for XRF base 1.5 / ERF / URF – with bootloader3d.

All you need to do is select the function required, buy the appropriate sensor or actuator and combine it with an XRF, perhaps conveniently using one of our boards. Using the XRF Configuration Manager (XCM) software you then program the XRF with the chosen personality and you are done.

Instructions on how to update the firmware on an XRF are here: http://openmicros.org/index.php/articles/84-xrf-basics/111-firmware-updating-how-to

Pre-defined sensor personalities

The following pre-defined sensor personalities are available

- Temperature sensor, thermistor
- Temperature sensor, Dallas
- Button sensor
- Analog
- Light sensor, LDR
- Generic
- PWM
- Ultrasonic
- Count
- Tacho

Pre-defined actuator personalities

The following pre-defined actuator personalities are available

- Relay
- RGB encoder

Communicating with sensors / actuators: LLAP

All Ciseco provided personalities use LLAP as the radio communication protocol to and from sensors and actuators. LLAP is a lightweight automation protocol in which sequences of 12 characters are exchanged.

Each message has the following format: aXXDDDDDDDDD where

1. "a" is lower case and shows the start of the message
2. XX is the device identifier (address A-Z & A-Z)
3. DDDDDDDDDD is the data being exchanged.

Note: Only the first “a” character is lowercase, the remaining message always uses uppercase.

The nice thing about this protocol is that the messages are understandable by ordinary people. For instance, if you wanted to know the temperature at a sensor with identifier RV, you’d send aRVTEMP----- (the dashes are padding) and you’d receive the following reply: aRVTEMP+23.1

If you want to know more, please read the LLAP pages on OpenMicros (http://openmicros.org/index.php/articles/85-llap-lightweight-local-automation-protocol). Start with LLAP - A simple elegant way to talk to objetcs and then move to LLAP starter before reading the other articles.

http://openmicros.org/index.php/articles/87-llap-lightweight-local-automation-protocol/llap-devices-commands-and-instructions

Creating your own sensor / actuator personality

Any developer worth their salt wants to make sensors and actuators do something a bit different from the pre-defined Ciseco provided personalities. So we got loads of requests for customisation, which we found hard to respond to (due to the volume in demand).

Allowing anyone to create their favourite sensor personality and allowing these to be uploaded to the XRF proved too painful. Programming the CC111x chip is a professional job and the increased chances of end user software interfering with the correct operation of the radio would mean our products would soon be seen as unreliable.

We needed a solution which allows you to program the personality yourself without compromising the reliability of the radio. Our solution is to add a separate end-user programmable Arduino processor to our radio unit in serial pass-through mode. This combination is now available in two form factors:

- RF?-328 combines an Arduino 328 with our SRF on the same Xbee form factor that is used by the XRF
- XinoRF places our SRF on an Arduino Xino compatible board

RF?-328 is ideal for low power deployment scenarios, whereas XinoRF is more suitable for development. Both units are software compatible, so software developed on one can be run without change on the other.

Now you can write your own specific sensor personality software, targeted on the straightforward Arduino platform and easily deploy it.

If you want you can create your own protocol, implement one of the industry standards, or you can follow LLAP for end-user friendliness and interoperability with Ciseco pre-defined personalities.

Conclusion

I hope that the above has given you the confidence to produce a successful project once you have received your kit. The links to the documentation on http://openmicros.org/index.php/articles point to more detail that you will want to follow up as you develop your project.

You may well want to check out our Wireless Inventors Kit for the Raspberry Pi: the easiest introduction to wireless control systems.