Remote Central Heating Control

I suppose the idea of trying to build a system to turn my home central heating system on & off via the Internet, came from seeing the very useful & responsive heating implementations using 'Nest', 'Hive' and similar.

However, I did not require any elaborate features such as detecting when I am at home or what my preferences are at specific times of the day. Admittedly these may be very important for some homes.

No, just a way to turn the heating on or off remotely and to set a specific temperature. Maybe, at a later date, modify the system to adapt to the external weather conditions.

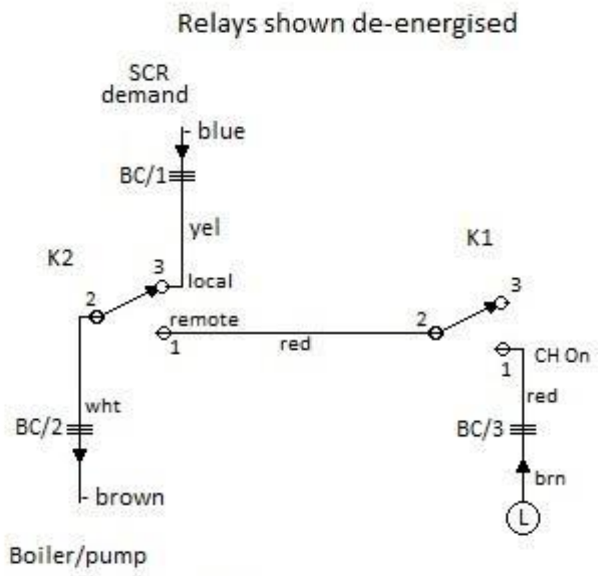
The design would use the existing home heating system, simply providing an option to switch between the current (wireless remote) and remote control via the Internet.

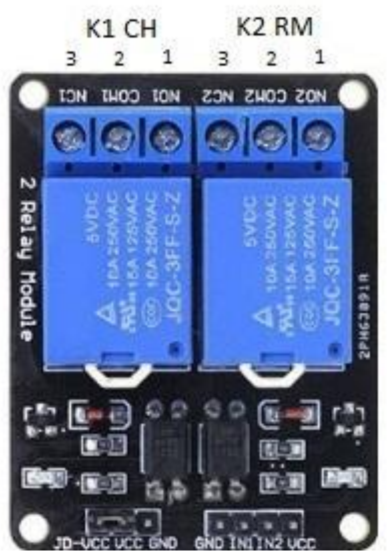
THE HARDWARE

So, starting with the hardware aspects, the control must be able to disconnect the (wireless) control demands and 'switch' to remote demands received.

The switching would be done using a relay units, these are readily available as modules in groups of 1 to 8. A single dual module was chosen, where one relay would switch from the home wireless control to remote (Internet) control and the second relay would switch the boiler, pump & rotary valve together.

The home wireless controller measured the room temperature (which ever room it was in) and compared it with the setting for that day & time. The decision to turn on/off the heating system is transmitted to a small receiver unit named the 'SCR', which in my case, was located near the system wiring panel (...in the 'airing cupboard).

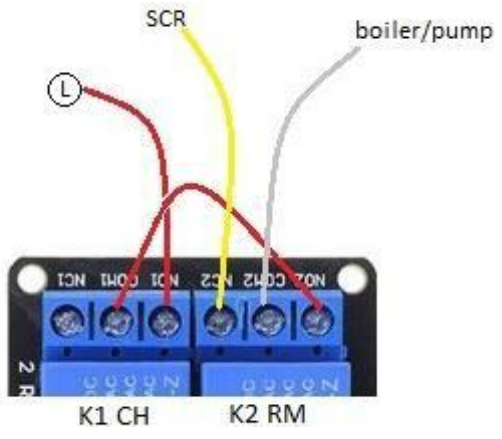




Under normal (ie local) operation, the mains Live feed would be connected directly to the boiler/pump line. When the temperature was below the required setting this supply would turn on the boiler/pump and hence the heating system.

The first change was to disconnect line from the SCR (ie wireless receiver) and connect this to the relay to be labelled 'K2' - see the diagram below.

'K2' would route the heating demand to a single contact of relay 'K1'. This relay now forms the heating ON or OFF operation.

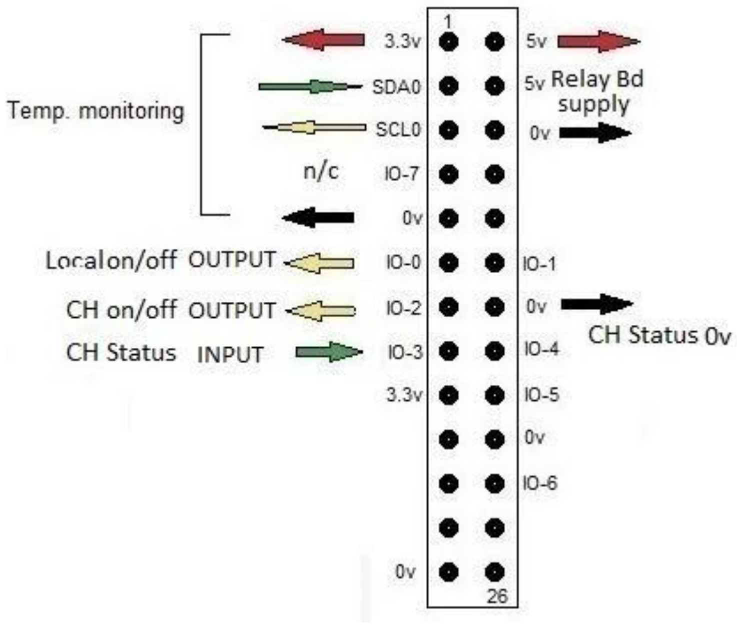


Relay module connections

All that is required is an energising current from a suitable controller. For many reasons I chose a Raspberry Pi (3) which had the input & output drive capability to operate the relays. In addition it has I2C capability and can be easily interfaced with a 'smart' sensor to measure temperature within the home.

The next step was to source a suitable 'I2C' temperature sensor. There are many that would suit, however I opted for a 'TMP102' sensor. Selected for its simplicity and the fact that I had used it before in a similar configuration.

The temperature sensor is connected to a the Raspberry Pi (RPi) via the usual four wires: Vcc, Gnd, SDA & SCL. In this instance the RPi GPIO was connected as shown below.



Most are self explanatory, the sensor input and supply (N/B 3v3), output to energise K2 labelled 'local on/off', output to energise K1 labelled 'CH on/off' and final an 'input' so far unmentioned.

This input is feedback conveying the state of the boiler/pump line. Without this, there would be no confirmation of central heating state (labelled 'CH status').

The CH status was monitored by connecting a low voltage mains transformer (240/6v) across the supply to the boiler/pump. I accept this does not truly confirm that the boiler/pump are doing their job correctly, but at least it confirms that they are receiving mains power!

This low voltage supply is (rectified and voltage limited) applied to one of the RPi inputs.

THE SOFTWARE

So, we have a temperature input, a boiler/pump input, a remote control demand output and a boiler/pump - on/off demand.

The algorithm was:

1. Using a 1min repeating loop the following actions are to occur:

2. On receipt of an remote demand 'ON', IO-0 is set high. This operation energises K2 and so remote operation is *enabled*.

2. The I2C bus receives I2C temperature data which is converted to a numeric value, representing the current room temperature.

3. Target temperature and current temperature are compared, as a result the state of K1 will be set. [ie If 'target' temperature < set temperature then CH on/off IO-2=high and K1 will energise]

4. If K2 - 'remote operation' has been enabled then boiler/pump will be connected to the Live mains feed and the heating system will turn on, otherwise control will remain to be from the wireless home controller.

The program code was written using Python and the code module was executed every minute by a 'CRON' job.

THE INTERFACE

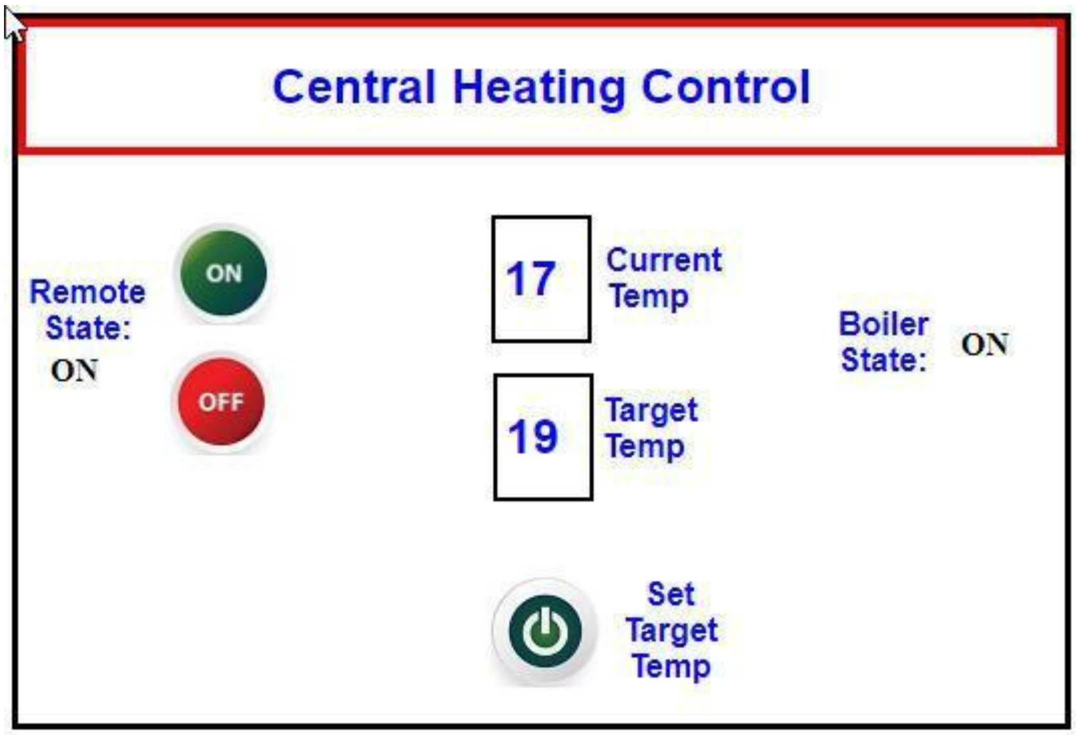
The remote ON/OFF operation (ie 'K2 state') and the required temperature (ie 'target') must entered via a web page interface.

Similarly the current room temperature, the target temperature and the boiler state need to be displayed.

A web server was already operational (another Raspberry Pi project) so a web page was created to handle the flow of data to/from the heating controller.

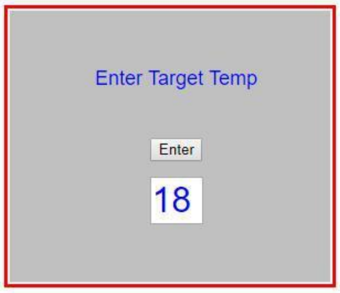
A combination of HTML and PHP code was used to read & display the boiler/pump state and the current room temperature and to accept a numeric value representing the target temperature.

Text files were used to receive/pass these parameters from/to the web interface to the Python control code.



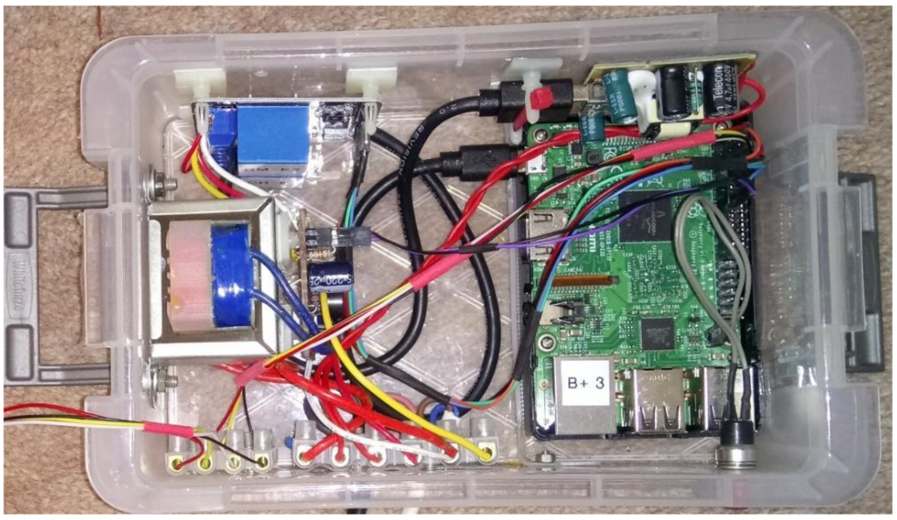
Notes: 'Remote State' = 'ON' : K2 = energised

'Current Temp' < 'Target Temp': K1 = energised .... boiler/pump = ON



Selecting 'Set Target Temp' will show:

To operate : Enter the required room temperature a and select 'Enter' .



Controller assembly ready for testing