I chose this protocol because I was able to find good descriptions (both on- and off-line), and because I had four Philips remotes of different vintages to test with. Two of them are standard RC5 and the other two are extended RC5 remotes.

An RC5 control "word" is made up of 14 bits. Two start bits, a toggle bit, 5 address and 6 data bits. The start bits are there to "wake up" the receiver and for adjusting the AGC circuit. The toggle bit toggles every time you release a button and press another or the same button. It is there so the receiver can detect if you are selecting "1" or "11" for instance. The first time you press the "1" button the toggle bit may be zero, and the next time it's a one. That way the receiver can detect that you have released the button and that you are pressing it again.

The 5 address bits are used to distinguish between different types of equipment. TVs use address 0, VCRs use 5 and satellite receivers use 10 for instance. The circuit includes a six-pin header, to enable you, the user, to select the address of the circuit. That way if you have a Philips TV, you can set your amp to use address 5 (VCR) to avoid interference.

The six data bits means that you can make remotes with up to 64 buttons (2^6), which should be enough for this purpose... But it wasn't enough for Philips for some reason, which is why they developed extended RC5. In that standard the second start bit is replaced by a high/low select bit, used to switch between two banks of 64 (possible) buttons. All the standard functions are in the lower bank (standard RC5), so I have chosen to ignore the high/low bit in my software.

The individual bits are all the same length. The two states 0 and 1 are encoded as level transitions. A "0" is encoded as a high to low transition, and a "1" as a low to high transition. The data produced by the encoder chip in the remote is modulated with a 36kHz carrier with a duty cycle of 0.25-0.33 (to save battery power). The receiver module in my circuit de-modulates the signal, so I have the original logic signal available on the output.

My software reads the code by first measuring the length of the start bit. If it's outside the expected range (the result of sunlight, a different remote etc.) an error flag is set, and the rest of the code is discarded.

After that I wait until the second half of the first address bit, and sample the input to read in the bit. Then I wait for the next bit and read the input again etc. Each bit is read twice for some extra error detection. When all the bits have been read, the circuit changes the required setting if the address and data bits match one of the patterns looked for, and if no errors occurred.