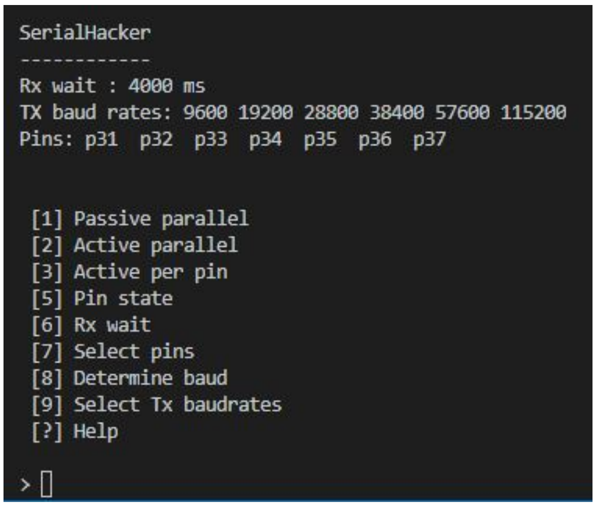
The main menu looks something like this:



```
SerialHacker
------------
Rx wait : 4000 ms
TX baud rates: 9600 19200 28800 38400 57600 115200
Pins: p31  p32  p33  p34  p35  p36  p37


 [1] Passive parallel
 [2] Active parallel
 [3] Active per pin
 [5] Pin state
 [6] Rx wait
 [7] Select pins
 [8] Determine baud
 [9] Select Tx baudrates
 [?] Help

>
```

1. Passive Parallel

This function will count high/low state changes on all the listed pins for the duration of **Rx wait.** (changeable using option [6])

The purpose is to find pins with activity to investigate further. If you scan pins, ensure they are connected to something or de-selected as floating pins are likely to generate false positives.

The output it the number of state changes per pin during the listening time. All the pins are checked at the same time. Its not expect that this count is 100% accurate, nor does it need to be.

At the end or the run, the active pins will be remembered, and you will have the option to try and determine the baud rate on each of them.

Active scans

The Active component here it the sending of data on a pin.

2. Active Parallel

This function will send a pre-defined wakeup signal on one pin at a time from the list, and listen for changes simultaneously on all the remaining pins. It will work through all the combinations. The exact wakeup signal needs to be added to the code as this is difficult to enter. Also, you need to have defined the board type you are using in the BEGIN USER DEFINITIONS section of the code. This defines the speed the data is transmitted at for different board types. Teensy, Due or Arduino have been configured.

If you want to use a different board, there is a hidden menu item that can be activated by entering ‘t’ (for Test pattern or Timing) from the main menu. This will send a 1010 binary sequence repeatedly on the first pin on the list. This allows you to tune delays in the send subroutine. You will need to connect the Tx pin to something like an oscilloscope to make these adjustments. The menu is hidden as this is an infinite loop.

At the end or the run, the active pins will be remembered, and you will have the option to try and determine the baud rate on each of them.

3. Active per pin

This function will send a pre-defined wakeup signal on one pin at a time from the list, and listen for changes on one of the remaining pins. All pin combinations and baud rates will be tried, so it can take some time.

At the end or the run, the active pins will be remembered, and you will have the option to try and determine the baud rate on each of them.

5. Pin state

Show the current high/low status of the selected pins.

6. Rx Wait

Set the time in milliseconds that the system will wait and listen for activity before moving onto the next task. You may not see any characters as you type, just go for it!

7. Select Pins

Allows pins to be selected or de-selected.

8. Determine baud

It will listen to the first pin on the list to help determine the baud rate. The output shows different baud rates and a counter that indicates the likelihood of a particular rate being used. Its not perfect, but accurate enough to make a very good guess.

There are a few limitations:

- Firstly, it assumes at least 20 high low transitions whilst listening.
- The data is fairly random. (not lots of repeating 1's or 0's) If this is serial ASCII data, it should be a fair assumption.

It works by measuring the duration (period) between each transition and compares that to known baud rate periods. If it decides the period matches 9600 baud, it will increase the counter for that speed to indicate that one bit period for this speed was found. If the data is random enough, this seems to work well enough.

An Arduino Uno seems to max out at 57600 on the current implementation, anything faster and it will not give correct results.

9. Select Tx baud rates

Allows the bauds rates to be used for stimulation whilst performing an Active Scan to be selected.