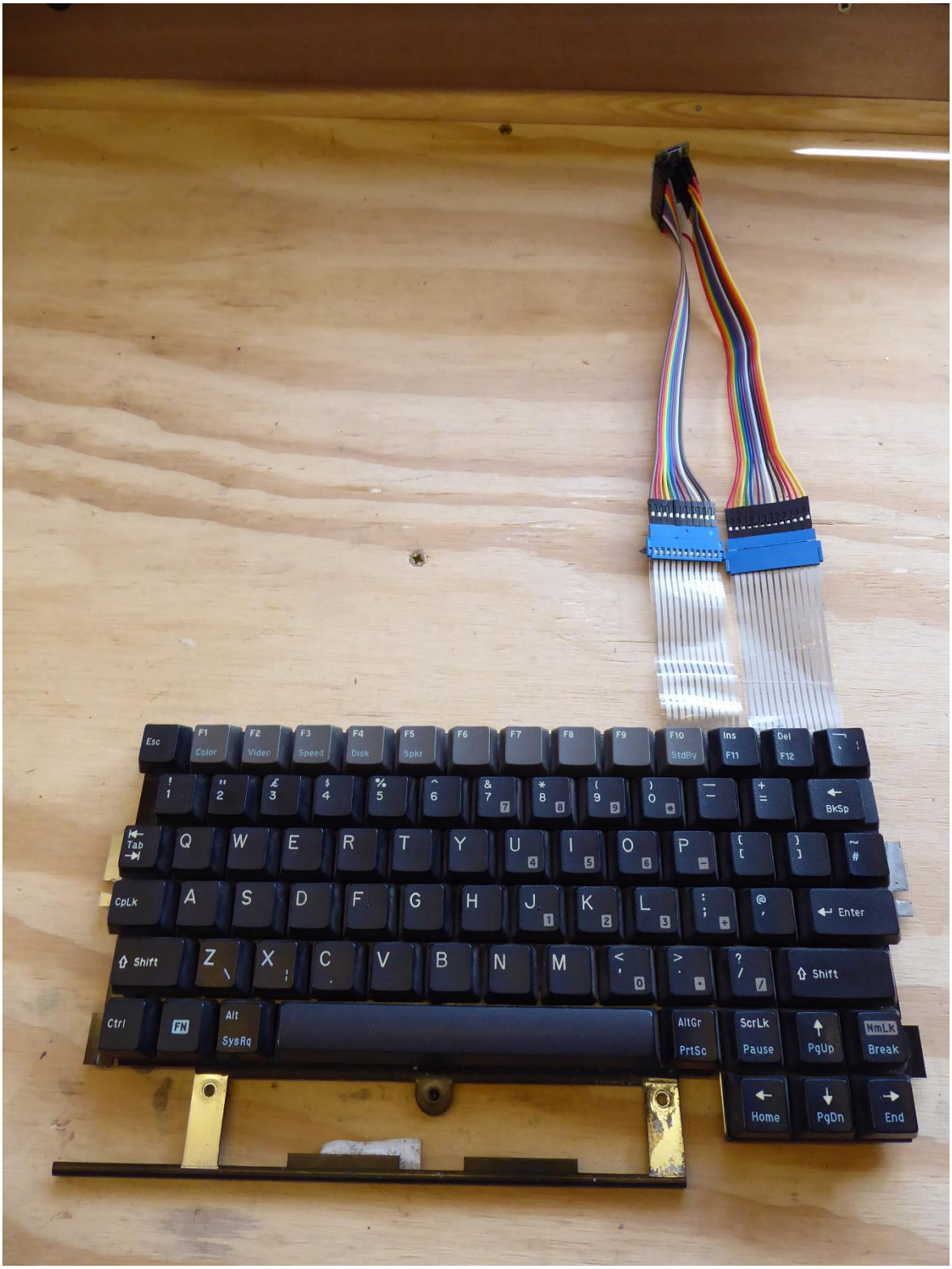
The GRiD 1550 keyboard shown below has a UK layout. The row and column cables from the key matrix have connectors that accept push pins from a standard ribbon cable. The ribbon cables have been soldered directly to the Teensy I/O pads, making sure to avoid I/O #13 (the LED control).



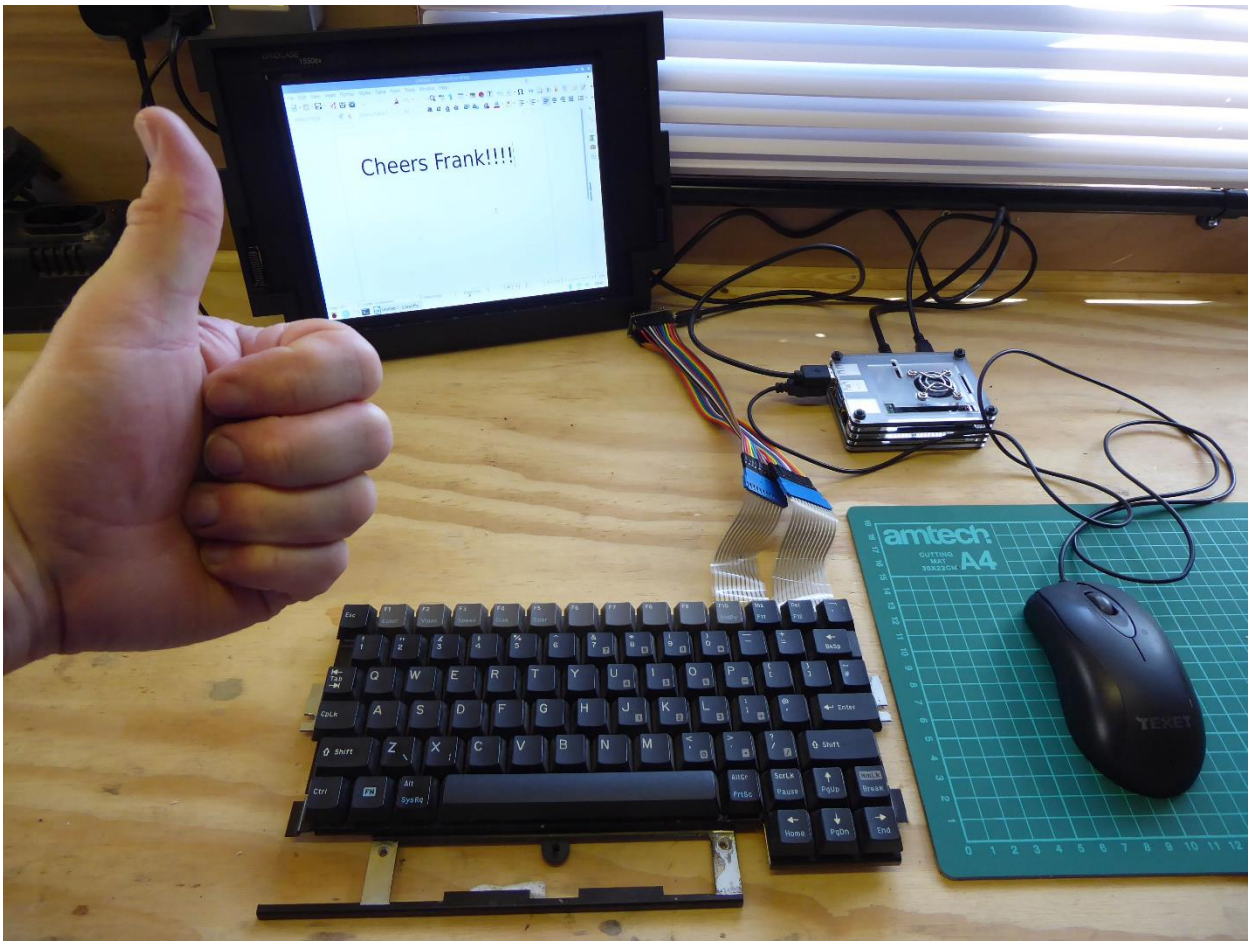
Running the matrix decoder code provided the 11 x 13 key matrix which was converted to the Teensy LC I/O pins, as shown below.

I/O#	23	22	24	21	25	20	19	18	17	16	15	14	12
1						F1	F2	F3	F4	F5		F9	
2	LSHFT	FN	RSHFT		Space								LALT
3						esc	1		F6	F7	F8	F10	
4						tab	Q	W	2	3		6	
5						CAPLK	A	S	E	4	5	7	
6				LEFT		LCNT		Z	F	D	8	9	
7				DOWN		RALT	X	C	G	R	T	0	
8				NUMLK		SCRLK	V	B	H	Y	U	-	
9				TILDE		UP	N	M	J	K	I	INS	
10				BKSLSH		RBRACE	,	BKSP	L	DEL	O	=	
11				RIGHT		ENTER	SLASH	.	“	;	LBRACE	P	

There were some anomalies with this keyboard due to its UK layout. Compiling the Arduino code with the UK keyboard selection allowed most of the keys to function properly. The exception was the “#” key and the “\” key. When a KEY_BACKSLASH is sent from the Teensy to the host, it produces a # so the backslash was placed in the matrix where the # should be. There was no easy way to produce a backslash so the code was modified to watch for a backslash key press and then send a sequence of keys. Alt code 92 was used to send a \ to a Windows PC but Alt codes do not work in Linux. A separate version of Teensy software was created that uses Unicode 5c to send a \. Using Alt codes and Unicodes are not very pretty but it gets the job done.

The Teensy code was modified to add a numlock matrix that provides a number pad when the NumLock key is selected. The number pad keys are shown below.

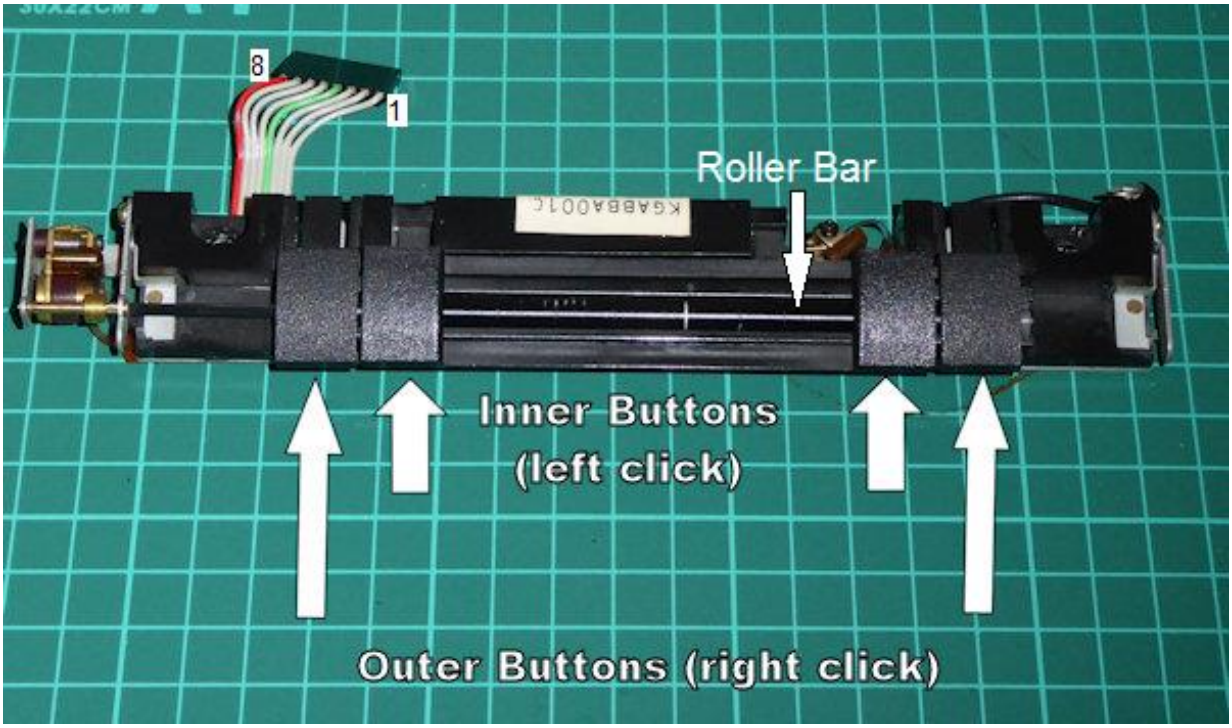
Here is the GRiD keyboard & Teensy connected to a single board computer that is driving the GRiD display.



The Teensy LC code for interfacing to a PC is called 1550_PC_Version.ino. The code for interfacing to a Raspberry Pi is called 1550_Pi_Version.ino. Both are at my GitHub Repo.

Nicely done Simon!

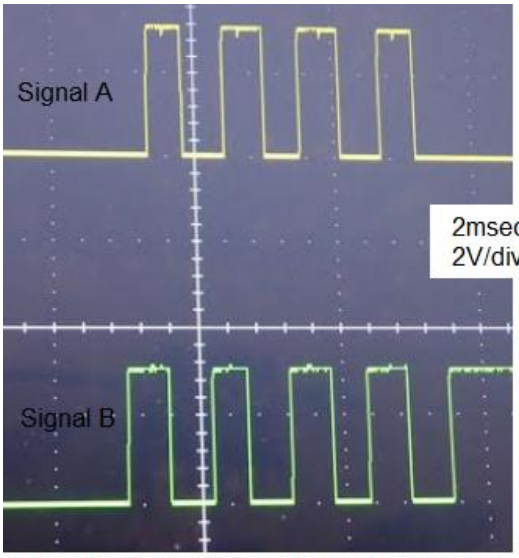
The GRiD 1550 laptop has a roller bar Mouse instead of a touchpad (see below). The user can spin the roller bar clockwise or counter-clockwise to move the cursor up or down. The bar can also be slid to the left or right a short distance from the center position to cause left or right cursor movement. The GRiD roller bar mouse uses rotary incremental encoders to convert up/down and left/right movement to electrical signals as described in this Wikipedia post.



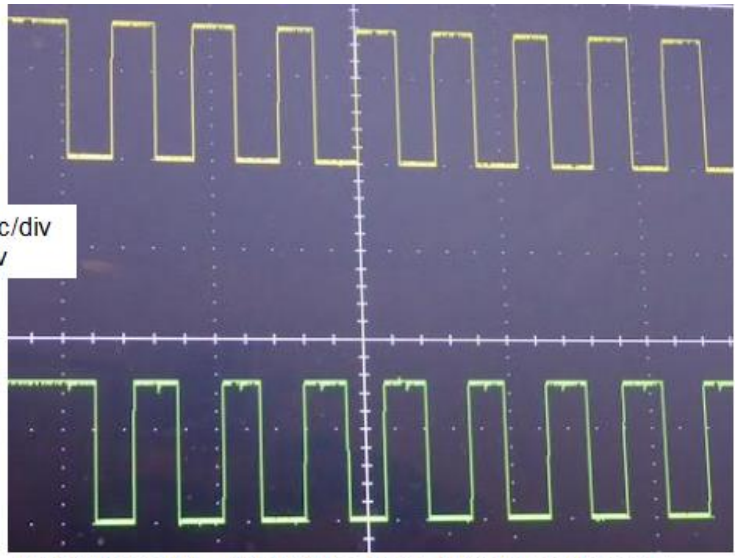
The connector to Teensy LC pinout is:

Pin #	Teensy	Function
1	Gnd	Ground
2	3.3V	3.3 volt DC power
3	I/O 3	Left/Right roller bar movement. Rotary encoder signal A
4	I/O 4	Left/Right roller bar movement. Rotary encoder signal B
5	I/O 5	Up/Down roller bar movement. Rotary encoder signal A
6	I/O 6	Up/Down roller bar movement. Rotary encoder signal B
7	I/O 7	Outer buttons (right mouse). Low when pushed. Driven high (no pull-ups needed).
8	I/O 8	Inner buttons (left mouse). Low when pushed. Driven high (no pull-ups needed).

The Teensy LC that is used to scan the keyboard did not have enough signals left to also decode the 6 signals from the roller bar mouse so a second Teensy LC was used. The Teensy code watches for the left/right signal A to change state and then samples signal B to see if A is leading or lagging B. A second set of A/B signals are watched by the Teensy for up/down movement.



Left Movement - Signal A lags B by 90 deg and has 8 clock edges.



Right Movement - Signal A leads B by 90 deg and has 17 clock edges.

A signed 8 bit counter is incremented when the bar is pushed to the right or decremented when pushed to the left based on left/right rotary encoder signals A and B. A second 8 bit counter is incremented when the bar is rolled up or decremented when rolled down based on up/down rotary encoder signals A and B. Every 24 milliseconds, the Teensy converts the count values to mouse commands and sends the results over USB to the host. The counters are then cleared for the next sample period.

The number of clock edges determines how much to change the two counters. A one to one ratio (i.e. one-edge to one-count) causes very slow movement. The code samples Teensy I/O pins 0 and 1 to let the user adjust the ratio in order to give 1 to 1, 1 to 2, or 1 to 3 cursor movement (see table below). These 2 I/O pins are set as inputs with internal pullups so if left unconnected, they produce a logic High. If using a regular mouse, the user may want to disable the roller bar mouse (by sending pins 0 & 1 low) so inadvertently touching the bar won't move the cursor.

I/O 1	**I/O 0**	**Result**
High	High	Count by 1
High	Low	Count by 2
Low	High	Count by 3
Low	Low	Mouse disabled

The Teensy LC code Grid_Mouse.ino is at my GitHub repo.