DIY Precision Slot Car Lap Timer

Simon Gardner - holisticsystems.co.uk - 28th December 2010

Introduction

This document describes how to build an electronic lap timer for slot car racing systems such as Scalextric.

It should work on any slot car system where the cars always go through the finish line in the same lane. For this reason, if you are using Scalextric Digital "cross overs", the current version of this lap timer isn't going to work for you.

This guide assumes you will be using the online slot car lap timing software available at http://holisticsystems.co.uk/scalextric-lap-timer

Overview

The system works by "hacking" a computer keyboard such that when cars pass through an infra-red beam they activate a circuit which simulates a key-press on the keyboard. The key-press is detected by the software and records the lap time.

The advantages of this approach are that no special driver software is required to talk to the device and the software should pretty much run on any computer that can run a web browser. (I have recently noticed that the software doesn't work on Internet Explorer, but this will be fixed in a subsequent version.)

All power for the circuitry is taken from the +5v USB wire.

Tools Required

- Soldering Iron
- Wire cutters
- Multimeter
- Screw driver
- Wire stripper
- "Helping Hands" very useful
- Glue gun very useful
- 5 volt power supply for testing (a spare USB phone charger is good for this)

Bill of Materials (Parts list)

Maplin product codes in (AANNA – price each)

- 1 x HC7404N integrated circuit - 6x inverters (UB03D - £0.79)
- 2 x 5v SPST Reed Relay (JH12N - £1.31)
- 2 x Infra-red Photo-diode Rx (CH11M - £0.69)
- 2 x Infra-red LED Tx (CH10L - £0.69)
- 2 x 22K resistor (M22K - £0.16)
- 1 x 39 ohm resister (M39R – £0.16)
- 2 x npn transistor (QQ15R - £0.22)
- Matrixboard approx 10cm x5cm should be enough (e.g. JP54J - £1.46)
- ABS Plastic Box (e.g. KC93B - £1.99)
- A spare piece of track to attach the diodes and "timing gantry" (£6.99!! from scalextric.com)
- Materials for a "gantry" to hold the photo-diodes above the track. I used a cardboard cereals packet. Next time, I would probably use something more substantial like plywood. (£??)
- A USB keyboard (could also work with old PS2 keyboard). A keyboard that has been damaged by say a coffee spill would be fine since the actual keys will not be needed. (approx £5)
- Approx 2 metres thin blue wire, 2 metres thin black wire, 1 metre thin red wire. (£2?) (Choose your own colours, but it's easier to "debug" if you use different colours for + and - . I doubt you'll use all 5 metres, but it would be annoying to run out :)

Approximate total cost £12.54 + cost of usb keyboard + gantry + piece of track = maybe £25?

Step-by-Step Instructions

These instructions are in 3 parts. Hacking the keyboard, Wiring up the controller circuit and wiring up the gantry.

Hacking the keyboard is the trickiest part - and whilst I have a working solution, I think my approach can be improved upon.

Wiring up the controller circuit and the gantry are really straightforward electronics tasks although as with anything like this, it's easy to make mistakes and you may need to get your multi-meter out to troubleshoot.

I've included some notes along with the circuit diagram to help with any troubleshooting. See below.

Once you've completed the 3 sections, connecting the 3 parts together should be extremely simple.

Hacking the Keyboard

part a) (a little tricky)

1. Dismantle the keyboard to extract it's controller. Take care not to damage any of the wires. The keyboard I used had a 2 layer plastic membrane inside with a network of thin "wires" connecting to the controller chip / usb cable. I think this is a pretty standard design.
2. The software registers a new lap when either the Q (for track 1) or the A (for track 2) keys are pressed. Identify where on the membrane corresponds to the positions of these keys.
3. If you think about it, you'll notice that the wires are arranged in a matrix with rows being on (say) the top layer and columns being on the bottom.

4. The wires end up being separated by a small air gap due to the thickness of a 3rd plastic sheet with small holes in between the two. Pressing a key forces the two layers together to make a contact. The keyboard controller detects which key has been pressed by detecting a closed circuit between the selected row and column.
5. Trace the rows and columns for the Q and A key back to the pins on the keyboard controller. Maybe make a note (say Q+, Q-, A+, A-) on the membrane indicating which pins, to remind you later. A good tip I've ready about doing this is to take a digital photo of the membrane and use a “Paint” program to paint individual wires to make this easier, however it's simple enough to do by hand if you concentrate.
6. On my keyboard (and I suspect most) the Q and A keys are on the same row and adjacent columns. This means that you will end up needing to label and wire up 3 pins from the controller. Q- and A- will probably go to the same pin. The actual polarity is irrelevant, it is the existence of a closed circuit which is detected.
7. On the keyboard I used, the controller connection was via an “edge connector” with about 16 pins. It proved to be quite tricky to get a robust connection between the edge connectors and the wires going to the relays.
8. In the end, I used a thin sheet of plastic film cut to the same size as the edge connector. I made some small holes corresponding to the appropriate pins on the connector and threaded the stripped ends of the appropriate wires through the holes.
9. I used a tiny amount of super glue to fix them in place whilst not too much that it would interfere with the electrical connection to the edge connector.
10. I then glue-gunned the plastic film in place to the edge connector and used the spring from a clothes peg to ensure a good contact between the wires and the edge connector. I also glue-gunned the clothes peg to make sure it stayed in place.
11. Overall, steps 7-10 are the worst, most fragile part of the design (although it has so far held up well for about 6 months) and I would definitely try to think of something better if I build another timer.
12. You should end up with a keyboard controller with 4 (or maybe 3) 5cm wires securely attached to the edge connector with good electrical continuity.
13. At this point you should be able to plug the usb cable into a computer and by carefully touching the appropriate wires together generate a Q or an A key-press on your computer.

part b) (nice and easy)

1. The USB cable will consist of 4 thin wires. 2 power and 2 signal. Examine the PCB of the keyboard controller to determine which are the positive and negative power wires. You may need to confirm this using your multi-meter.

2. Attach some wires (5cm should be long enough) to appropriate points on the PCB. Use different coloured wires for positive and ground and be consistent in your application of this scheme. I used blue for positive, black for ground and blue/white for “signal” ie where the line is sometimes positive or 0.

Wiring up the controller Circuit

Not sure how much I need to explain here?

Place the components on the matrix board and ensure that the board has sufficient space to contain all the components whilst also fitting securely inside your chosen ABS box.

I found a couple of paper clips soldered to top and bottom of the matrix board were useful as a “power bus” for the 5+v and ground lines.

Wire up the components on the matrixboard according to the circuit diagram. It might help to lay the components out roughly in the same positions as the circuit diagram ie with the relays on the right, inverters on the left etc.

Take care to avoid shorts (obviously). This was my first significant electronics project and I ended up using electrical tape to mask the ends of the wires near the matrix board. This isn't very neat and I wished I'd thought to apply some rubber sheath on the ends of the wires. (Or maybe some other solution).

Wiring up the LEDs and Photo-diodes and making a gantry.

It's important that the photo-diodes go under the track and the LEDs are mounted on the gantry. (If you get it the other way around there is a chance the light from one LED could shine into the wrong photo-diode preventing a car from being detected on that track. The LEDS and photo-diodes look very similar, so it's important to keep track of which is which

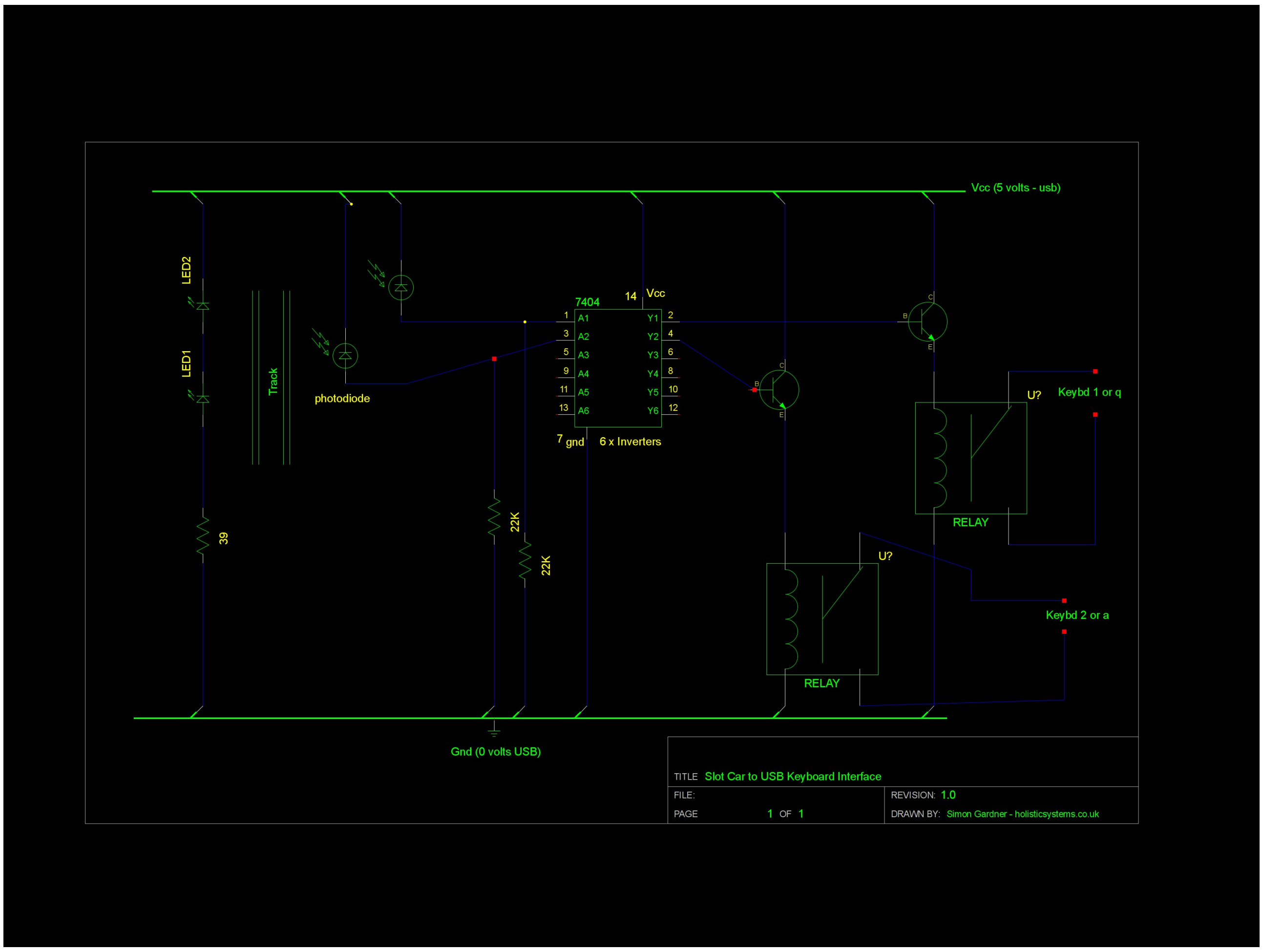
It's also quite important to make sure the active part of the photo-diode is mounted dead centrally between the metal rails to avoid the rails casting a shadow or even completely blocking light to the diode.

It isn't critical to ensure the LEDS are mounted *exactly* vertically above the photo-diodes as there is a good allowable margin for error. That said, there is no reason not to aim to get the LEDS accurately in position directly above the diodes, but no need to be obsessive about it.

I made my gantry out of a cardboard cereal packet cut into a strip about 6 inches wide and folded to raise the LEDS about 3 inches above the track. Over time the cardboard has become a bit floppy and needs to be propped up at the sides with a couple of books. It still works OK, but without the books, passing cars cause the gantry to wobble which can false readings on the timer. If I build another timer I will certainly build a much more substantial gantry out of wood (probably thin plywood) and screw it securely to the track.

I would use enough spare wire connecting the LEDS / photo-diodes to the control box so that you can place the control box somewhere where it is unlikely to be subjected to vibrations, impacts or direct sunlight. I had enough spare wire to tuck the control box under a bridge about 20 cms away. I wish I had a bit more “slack” though. I was concerned the relays might be accidentally triggered by vibrations, or the relatively fragile connection to the USB keyboard controller might get damaged. Especially if strong sunlight heated the box and warmed the glue. Fortunately, I've had no problems so far.

Circuit Diagram



Notes:

1. The LEDs are wired in series with a small resistor required to limit the current. The value of the resistor will control the “brightness” of the LED - although obviously, you won't be able to see anything because it's infra-red. A 20 ohm resistor will give the maximum brightness possible whilst not exceeding the operating current of the devices. The suggested value of 39 ohms worked well for me, but your results might vary. You might be tempted to wire the LEDS in parallel, in which case you'll need to recalculate the value of the balance resisitor - and it will need to be quite a bit higher to prevent burning out the LEDS.

2. When the device is on and the beams are unbroken, the photo-diodes are conducting making the input on the inverter to “on”. The output from the inverter is off, the transistor is off, the relay is off and the keyboard controller doesn't detect a keypress.

3. When the beam is broken, the photo-diode stops conducting. The 22K pull down resistors allow charge to drain from the input of the inverter allowing it to go “off”. This makes the output from the inverter go “on”, the transistor turns on, the relay turns on and the keyboard controller “thinks” that the appropriate key has been pressed.

4. It's possible that the HC7404 could provide enough current to drive the relay but I decided to play it safe and drive the relay via a transistor.

5. **Note the pin connections on the HC7404 should be as per the numbers next to the pins.** Not laid out like the drawing of the chip implies. ie you should use pins 1,3 for input and 2,4 for output.

Disclaimer

This documentation is provided as is with no warranty whatsoever. By attempting to build this project you automatically accept that you assume any and all responsibility for the correct and safe operation of the device.

In all likelihood nothing is going to go badly wrong and the world isn't going to end but the bottom line is you need to know what you're doing and exercise your own skill and judgement in following these instructions and take all responsibility for the end result.

Have fun !

Licence