The Pinscape Controller

A How-To Guide

M. J. Roberts • August, 2014

Pinscape is the name of my virtual pinball cabinet, which I've been building for about nine months (and counting). The Pinscape Controller is a device I designed as part of that project, to handle two special forms of input peculiar to pinball machines: the plunger, and nudge sensing. This guide explains how to build one of your own.

Overview

The Pinscape Controller is a USB device that you plug into the PC in your virtual pinball cabinet. It emulates a USB joystick, so Windows recognizes it automatically without installing any drivers. It also works seamlessly with Visual Pinball, which has built-in support for using joystick input for nudging and plunger position sensing.

Physically, the main controller runs on a KL25Z microcontroller board. This is a credit-card sized single-board computer. It has an on-board accelerometer that we use for nudge sensing, so it has to be mounted flat on the floor of your cabinet, and secured in place so that it moves with the cabinet. It connects to the PC/motherboard via a USB cable, which also supplies it with power.

The plunger sensor is a separate device, called a CCD linear array, which is packaged as a small, self-contained circuit board, about 3" x 1/2". This must be mounted adjacent to the mechanical plunger, and electrically connected to the KL25Z. You'll also need to install a small light source – one or two 20 mA LEDs seem to work well for this purpose, mounted near the floor of the cabinet directly below the plunger. These will of course require wiring for power.

You might also need to build some external circuitry, depending on which set of features you end up using. And of course there will be wires between the various pieces.

Features

The controller has four main functions. The features work independently, so you can use any subset.

- Plunger position sensing. The software works with a CCD sensor to read the position of the plunger. As you move your mechanical plunger back and forth, the on-screen plunger in Visual Pinball tracks its motion.

- Nudge sensing. The controller uses an on-board accelerometer to detect cabinet motion. Visual Pinball has built-in support that applies physical acceleration readings to its simulation, so the ball on screen reacts when you nudge the cabinet. This is true analog nudging, not the simulated type you might be used to from the VP keyboard interface. With accelerometer input, a little nudge yields a little reaction, a bigger nudge yields a bigger reaction, and nudges are fully directional.

- Button input wiring. You can wire up to 24 pushbuttons and switches to the controller for purposes like flipper buttons and the Start button. The PC sees these as joystick buttons, which VP knows how to read for its input controls.

- LedWiz-compatible output control for feedback devices. This is a bonus more than a feature; it has some significant limitations and requires considerable soldering work. But it might come in handy if you need just a few extra ports. (That's my situation and my motivation for including this feature.)

The competition

There are two commercial products that offer similar features to the Pinscape Controller. One is the VirtuaPin Plunger Kit; the other is the Nano-Tech Mot-Ion Controller. The VirtuaPin kit is the newer of the two, and I get the impression that it's easier to set up and more robust. Both kits provide analog plunger input, nudge sensing via an accelerometer, and wiring for button inputs. Both are finished, professionally built, nicely packaged products that you can use right out of the box. They're both reasonably priced, and in fact might save you money compared with my design. Another important difference: the kits come with customer support; mine comes with a "No Warranty" disclaimer. You should take a look at the commercial kits before you proceed.

The reason I didn't use one of those kits myself is that I wanted more precise plunger sensing. Let me be clear: for most tables, the kits are perfectly good in this respect. They do the basic job of letting you pull back the plunger and release it to launch the ball. That's all you need on the majority of tables. But I wanted more precision, for two reasons. One is animation quality: with the kits, the on-screen plunger moves in visible steps, less smoothly than I'd like. That's of no practical importance; I'm just picky. The other, practical concern is skill shots. Some tables have plunger skill shots that require a rather precise touch, and the kits I tried didn't offer enough control (in my opinion) for a fully convincing simulation. This too is a small thing, but there are some skill-shot tables that I'm particularly fond of.

I want to emphasize that the commercial kits are good options that most people would be perfectly happy with. They'll save you a ton of work (and possibly some money) compared with building your own. Unless you're as picky as I am about the same details I mentioned, you might never notice the

performance quibbles that made me reluctantly give up on the kits. There's also some risk that you won't be able to get my design working properly – I've tried to thoroughly document how I built mine, but I'm sure I don't have it entirely down to a science. I'm also assuming some knowledge of electronics. You should consider these factors before embarking on building your own.

No Warranty

Just to be sure there's no misunderstanding, I want to spell out that this documentation and the associated software has NO WARRANTY of any kind. I'm making it available at no charge in the hope that it will be useful, but I can't guarantee that it will work or that you'll be successful building your own version. I've tried to accurately describe what I built, but I might have missed important details or made errors, and the plan hasn't been tested by anyone else. Proceed at your own risk.

THIS DOCUMENTATION IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE DOCUMENTATION OR THE USE OR OTHER DEALINGS IN THE DOCUMENTATION.

Parts List

There are two main suppliers you'll need for parts. The first is an electronics distributor. If you don't already have a preferred vendor, Mouser Electronics and DigiKey are good on-line sources with mind-bogglingly huge selections. The second is a pinball parts store. You've probably already found one if you're building a virtual cabinet, but if not, try Pinball Life and Marco Specialties. VirtuaPin also offers some of the most commonly needed parts for virtual cabinets, often at great prices, although they don't have the kind of comprehensive selection the distributors do. You can also try eBay, although in my experience, used pinball parts on eBay are as expensive as new parts from the distributors, and discrete electronics are just too big a pain to find in the gargantuan eBay haystack.

Note that the specific part numbers and supplier links are only suggestions, to make shopping easier. If you're an electrical engineer, you'll probably scoff at my choices for the various components (and if so, let me know; I'd be happy to incorporate any corrections or suggestions for improvements).

Electronics required for all variations:

- Freescale FRDM-KL25Z microcontroller board
- USB cable, USB-A to Mini-B, 1.8m (or length as needed*)

Electronics for the plunger feature:

- TAOS TSL-1410R CCD linear array
- Blue LED, 2 each (or similar light source of your choosing)
- Molex .062 5-circuit plug ††
- Molex .062 5-circuit receptacle ††

Molex .062 crimp pins, 5 each ††
Molex .062 crimp sockets, 5 each ††

Electronics for the plunger calibration button (optional even if you install the plunger sensor):

Illuminated momentary pushbutton switch
2N4401 transistor (or equivalent NPN switching transistor)
2.2K 1/4W resistor
82 ohm 1/4W resistor
Molex .062 4-circuit plug ††
Molex .062 4-circuit receptacle ††
Molex .062 crimp pins, 5 each ††
Molex .062 crimp sockets, 5 each ††

Electronics for the LedWiz output feature:

8+8 vertical pin header for jumper J1 ††
8+8 wire housing for J1 ††
10+10 vertical pin header for jumper J2 ††
10+10 wire housing for J2 ††
Crimp terminals for the wire housings above, 1 for each output you plan to use ††
ULN2803A Darlington transistor array, one for each 8 outputs you plan to use
PC817 optocoupler, one for each output you plan to use
BUK9575-55A MOSFET, one for each output you plan to use
1N4007 diode, one for each output with an inductive load (contactor, motor, coil, etc)
220 ohm 1/4 watt resistor, one for each output
470 ohm 1/4 watt resistor, two for each output
Perforated circuit board, sized as needed, for building the output stage circuits

Pinball parts required only for the plunger feature:

Ball shooter assembly, Williams part no. B-12445-1**
Ball shooter mounting plate, Williams part no. 01-3535
Medium-low tension ball shooter spring, Stern part no. 266-5001-04†

Miscellaneous:

#10-32 x 3/4" machine screws, 3 each (for mounting ball shooter assembly)
#4 wood or sheet metal screws, 2 each (for mounting KL25Z to cabinet)
6mm nylon spacers (for mounting KL25Z)
22 AWG wire (stranded or solid; solid is a little easier to solder to the boards)
Solder

* The KL25Z connects to the PC via a USB-A to Mini-B cable. Choose a length as dictated by your cabinet's internal layout. It has to be long enough to reach from a USB port on your PC to the port on the KL25Z. 6' should be safe for most cabinets.

** You can also order the component parts of the plunger individually if you prefer. That lets you customize the shooter spring and knob style. Several colors and styles of knobs are available from the regular pinball suppliers, and third parties offer gimmicky themed rods for various tables. Search the Web for "custom pinball shooter" for some leads. If you want something truly custom, you can order a

knobless shooter rod from Pinball Life and epoxy a plastic doo-dad of your own design to the end. Highly recommended for fellow OCD sufferers.

† I recommend using a medium-low tension spring instead of the standard tension spring supplied with the full assembly. Your virtual plunger will never actually strike a ball, just dead air, which makes the standard strength a little harsh. Lightening it up a bit helps. I think it also just feels smoother with the lighter spring.

†† All connector specs are the gentlest of recommendations, to help you make your way through the overwhelming number of choices if you shop someplace like mouser.com. I rather like the Molex .062 series because it comes in a variety of pin configurations, so it's easy to make everything foolproof by using different connector types for nearby circuits – you can't accidentally plug the wrong things together because they won't fit. I first came across these plugs because they're widely used in Williams machines from the 80s and 90s. Most virtual cabinet builders seem to favor screw terminals for most connections; I much prefer pluggable connectors because they're so much faster to connect and disconnect when doing work on the machine. They also let you bundle groups of related wires together into a single connector, eliminating the need to trace wires. But connectors are just connectors; if there's another style you prefer, definitely go with what you know.

Tools

You'll just need the basics:

Soldering iron
Wire cutters
Wire strippers
Screwdrivers
Needle-nose pliers
Crimp tool (if desired, for the Molex crimp pins; needle-nose pliers are a workable substitute)

Update your KL25Z boot loader

There's a one-time setup step required when you first take your KL25Z board out of the box, to update it to the latest version of the boot loader firmware. Once you've completed this step, installing the Pinscape Controller software is easy. Here's what you need to do:

- Go to http://www.pemicro.com/opensda/
- Download the "OpenSDA Firmware (MSD & Debug)"
- Unzip the file
- Follow the instructions in "Updating the OpenSDA Firmware.pdf",
- Use the same procedure to install MSD-DEBUG-FRDM-KL25Z_Pemicro_vxxx.SDA (from the same zip file)

Attention Windows 8 users! When I got my KL25Z, the update process didn't work on Windows 8 or 8.1. The instructions simply didn't work, with no indication why. This turned out to be an incompatibility with the Windows 8 USB drivers in the factory-installed version of

the boot loader. The only workaround was to find a Windows 7 (or earlier) machine and do the firmware upgrade there. Once updated, the card works on Win 8, but you couldn't get the upgrade installed in the first place on Win 8. Hopefully they've fixed this in the factory firmware by now, but if the update instructions mysteriously fail on your Win 8 machine, try Win 7 or earlier. Make sure the new boot loader you install is dated June 2014 or later.

Download the controller software

The Pinscape Controller software for the KL25Z is an open source project, so you're free to use it as-is or to customize and extend it by modifying the source code. The software is available here:

http://mbed.org/users/mjr/code/Pinscape_Controller/

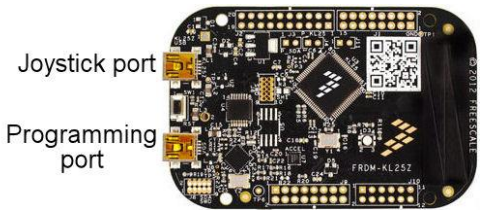
To get an installable binary version, follow the link above, then click the Build Repository button on the right side of the page. That will take you to a build page. Make sure that the Target Platform is set to FRDM-KL25Z, then click the Compile button.

Note: If you don't see the KL25Z in the Target list, you might have to add it to your account. To do this, go to the Platforms page, which you can reach from a navigation button at the top of the page. Find the FRDM-KL25Z in the list. Click on it to bring up the device page, then click the button "Add to your mbed Compiler".

The compilation process will take a minute or two. Once that completes, a Download button will appear at the bottom of the page. Click it and save the file. This will give you a file ending in .bin that contains the compiled binary version of the controller software. This is the file you install onto your KL25Z, using the steps in the next section.

Install the controller software onto the KL25Z

Plug the KL25Z into your PC using the programming port (if you hold the board with the ports at the bottom and the chip side facing you, the programming port is the one on the right – the same port you used to update the boot loader). The KL25Z should now appear in Windows as a disk drive. In this mode it acts like a memory stick with 128k of flash.



Now simply drag and drop the .bin file you got from mbed.org (see above) onto the KL25Z drive.

You can repeat this step at any time to update to a newer version or to do a "factory reset" on the software. Note that doing so will erase the plunger calibration and any configuration settings you changed, so you'll have to repeat those steps any time you reinstall the controller software.

Plug in the joystick port

Once you've installed the controller software, plug the KL25Z in via its other USB port – the one labeled Joystick Port in the diagram above. Windows should recognize the device as a USB joystick called "Pinscape Controller".

If you bring up the Windows control panel for "Set up USB game controllers", and open the Pinscape Controller entry, you should see the x/y axes of the joystick display reacting as you tilt and bump the KL25Z. This represents the accelerometer readings that the KL25Z is sending to Windows.

Note that it's perfectly fine to leave both USB ports on the KL25Z plugged in continuously. You don't have to disconnect the programming port when using the device as a joystick, and you don't have to disconnect the joystick port when re-installing the controller software. But there's also no need to leave the programming port plugged in all the time, if you don't want to tie up an extra cable and an extra USB port on the PC.

If you should ever feel the need to do a full power cycle of the KL25Z (because the software appears unresponsive or stuck, say), be sure to unplug **both** of its USB ports for a few moments. Each port independently provides power to the board, so you have unplug them both to cut all power and force a hard reset.

Status LED

The KL25Z has an on-board RGB LED, which the controller software uses to provide at-a-glance status information. The color/flash patterns are:

- Off: the device isn't plugged in or hasn't successfully completed the USB handshake with the host.
- Short red flash, about every 3 seconds: the host computer is powered down or in sleep/suspend mode.
- Two short red flashes, about every 3 seconds: the USB connection has been broken. This could be because the cable has been physically disconnected, or a communications failure occurred.
- Alternating red/green: the device needs to be reset due to a change to the LedWiz unit number. Unplug the device for a few seconds and plug it back in, or press and hold the Reset button on the KL25Z for a few moments.
- Alternating yellow/green: the device is connected and operating normally, but the plunger hasn't been calibrated. If you're using the plunger sensor feature, run the plunger calibration procedure.
- Alternating blue/green: the device is connected and operating normally.
- Flashing blue: the plunger calibration button is being pressed, but calibration mode hasn't startd yet. Keep holding the button to enter calibration mode (or release it to cancel).
- Solid blue: plunger calibration mode is in progress. Calibration mode lasts for about 15 seconds. See the section on plunger calibration below.

Software configuration

The KL25Z controller software is ready to use as soon as you install it. However, it does have a few configuration options that you can set, if you wish. You set these using the configuration tool, which runs on Windows and sends commands to the controller via the USB connection. Note that the controller needs to be plugged in to the USB port and operating in joystick mode before you can send it commands. The download link for the tool is on the mbed project home page.

Configuration changes are saved to the non-volatile (flash) memory on board the KL25Z, so your settings will survive power cycles, controller resets, PC resets, and USB disconnections. Reinstalling or updating the controller software, though, will clear memory and reset the software to factory settings.

The available options are:

- Enable or disable the plunger sensor. The sensor is enabled by default. You can leave this enabled even if you don't attach the sensor – the software will simply see random pixel values if no sensor is attached, and will otherwise function normally. However, it might increase the rate of accelerometer reports sent to VP if you disable the sensor, because it takes a certain amount of time to read the sensor input (even if it's not attached) on each cycle.

  Note that this probably won't affect the quality of the accelerometer data. The software samples the accelerometer at a fixed 800 Hz rate (the maximum hardware sampling rate for the accelerometer) regardless of the plunger sensor status, and averages readings over the joystick reporting cycle. So the same net readings should reach VP one way or the other. Even with the CCD enabled, joystick reports occur about every 30ms, which is below the threshold of human perception for latency.

- Change the LedWiz unit number. Each LedWiz device has a unit number, set in the firmware. The unit number lets the PC direct commands to the right device if you have more than one LedWiz installed in your cabinet. On a real LedWiz, the unit number is set at the factory; the default, if you didn't specify otherwise when you ordered, is unit #1. The Pinscape Controller presents itself by default as unit #8, but you can change this at any time using the configuration tool.

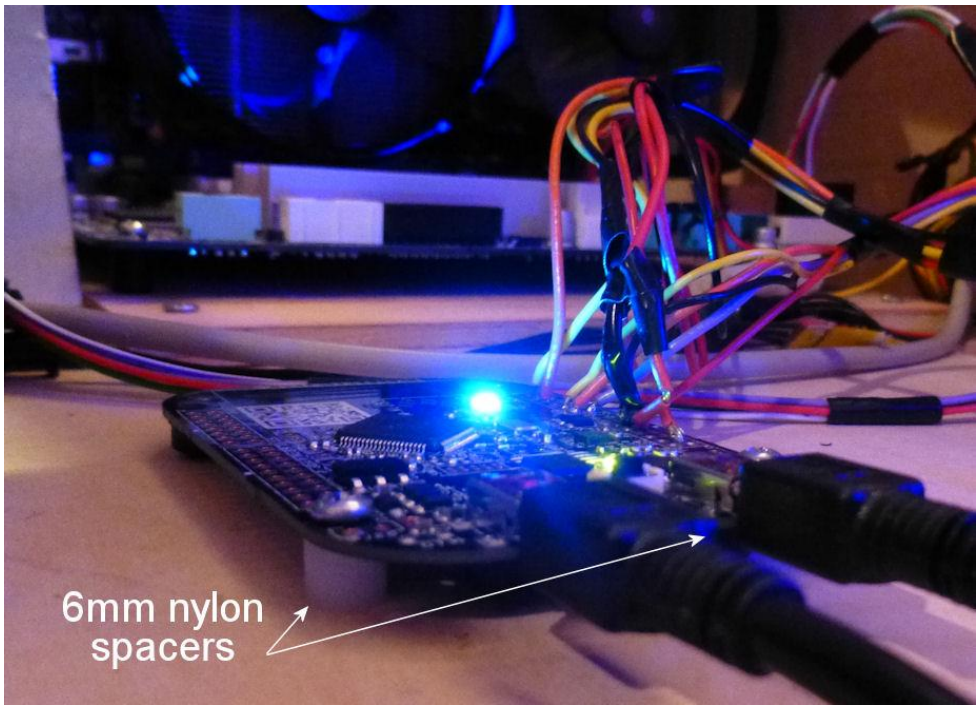
  If you change the unit number, you'll have to reset the controller before the new unit number takes effect. The diagnostic LED will flash red/green to indicate that a reset is needed. Simply hold down the Reset button on the KL25Z for a few moments, or unplug the USB cable for a few seconds and then plug it back in.

Mount the controller board in your cabinet

The KL25Z should be installed flat on the floor of your cabinet with the chip side facing up. The side of the board with the USB connectors should face the front of the cabinet. If this orientation isn't convenient, it's okay to rotate the board in 90 degree increments. If you do rotate it, you'll have to adjust the joystick axis settings in VP to match the rotation (see below).

Ideally, the KL25Z would be mounted perfectly horizontally, but this isn't practical because most pinball cabinets are slightly tilted. That's okay, though; the controller software automatically calibrates itself whenever it detects that the cabinet is at rest, so it will compensate automatically for the slope of the cabinet floor. So you should simply mount the board flat on the floor of your cabinet.

It's important to mount the KL25Z so that it's firmly fixed in place inside the cabinet. We want the accelerometer to experience the same accelerations as the cabinet during nudge events, which will happen as long as it's firmly attached so that it moves with the cabinet. The KL25Z has rubber feet pre-installed, and two holes (on the outside corners on either side of the USB connectors) for screws. #4 x 1" screws (for wood or sheet metal) are a good fit. For the best fit, use a 6mm-tall nylon spacer on each screw (under the board, between the board and the cabinet floor).



Configure Visual Pinball

Visual Pinball has built-in support for accelerometer and plunger input using the joystick, but it's turned off by default. To activate it:

- Open VP (in design mode; no table needs to be loaded)
- Select Preferences > Keys on the menu
- Check the box for "Enable Analog Nudge"
- Check the box for "Enable Nudge Filter", if present (see below)
- Under "Axis Alignment":
    - Set X-Axis (L/R) to "X-Axis" on the drop-down
    - Set Y-Axis (U/D) to "Y-Axis"

- Set X-Gain and Y-Gain to 1000
- Set Plunger to "Z-Axis"
- Set Dead Zone to 0
- Un-check the "Reverse axis" boxes for X, Y, and Z

The settings above assume that you installed the KL25Z in the standard orientation, with the USB connectors facing the front of the cabinet. If you used a rotated position, adjust as follows:

- 90° clockwise: swap the X and Y axes, and check "reverse axis" for Y
- 90° counterclockwise: swap the X and Y axes, and check "reverse axis" for X
- 180°: check "reverse axis" for both X and Y

The "Enable Nudge Filter" checkbox is new as of August 2014. This checkbox won't be there if you're using version 9.9 or earlier. If available, the nudge filter applies some special filtering to accelerometer readings to make the effect on the simulated ball more realistic. Without the filter, nudging with an accelerometer tends to leave the ball with an unrealistic amount of residual velocity. The filter corrects for certain systematic errors in the way the readings are passed to VP.

If you want the nudge filter before it's released officially, an unofficial modified version of VP 9.9 with the feature is available here:

https://www.dropbox.com/sh/0gtnrck3yr9w9oa/AAAZpK2Nhw2HtOshcJWkIe6ha

Look for an .exe file with "vpinball accelerometer patch" in the filename. This version doesn't give you an option to turn it off, so you won't see the checkbox there either. That folder also has the source code changes for the feature, in case you want to build the whole thing from source or patch it into a different version of VP.

Wire the plunger sensor

The wiring for the TSL1410R is shown in the schematic below. Several of the connections are simply between terminals on the sensor itself. There are five connections between the sensor and the KL25Z.

The manufacturer's data sheet for the TSL1410R is available here:

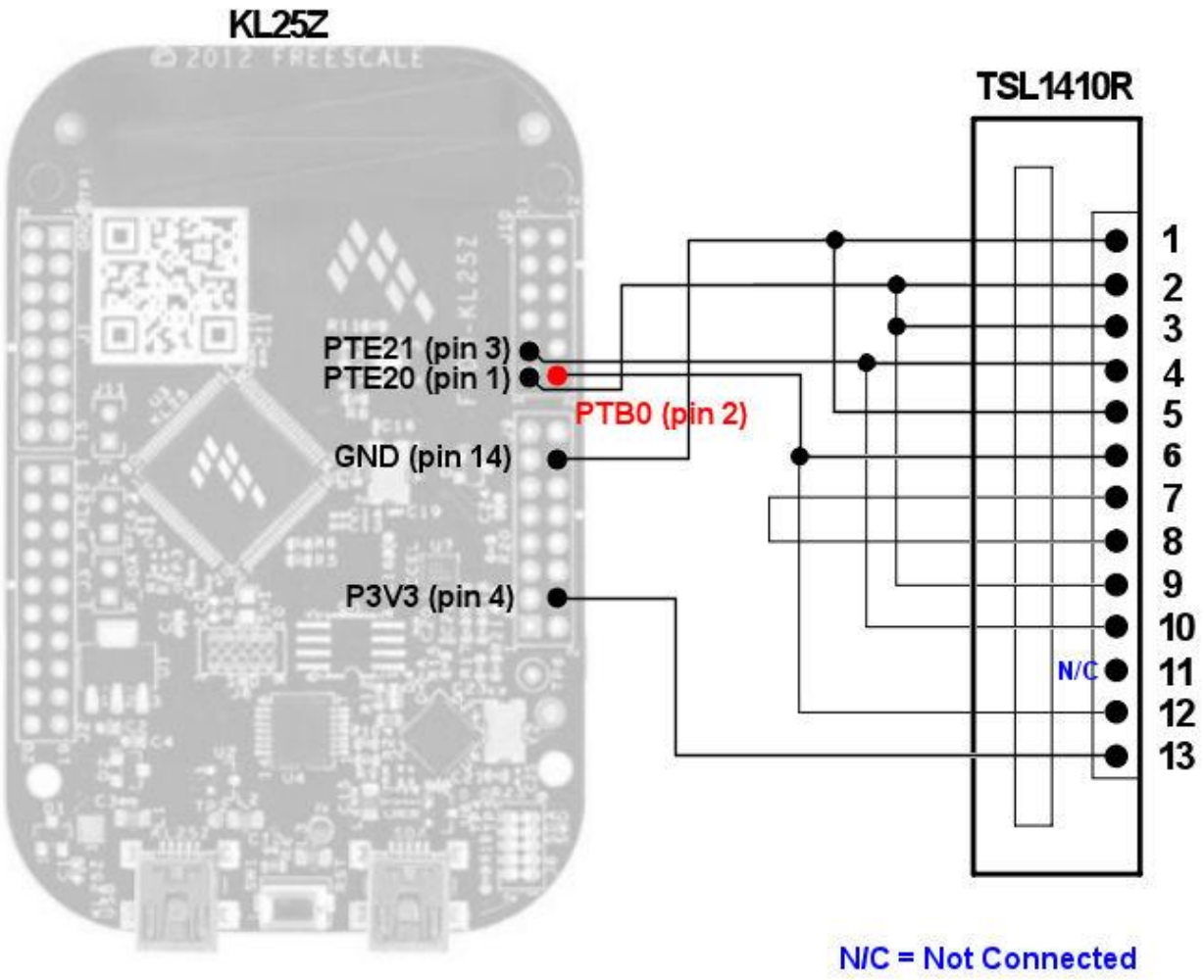
http://www.taosinc.com/getfile.aspx?type=press&file=tsl1410r-e29.pdf

I placed an in-line plug/receptacle pair (see the Molex connectors in the parts list) between the KL25Z and the sensor, so that either part can be removed from the cabinet independently. I soldered wires directly to the terminals on the sensor and the solder pads on the KL25Z.

I would have preferred to use a pluggable connector to attach to the sensor, but I haven't been able to find anything that fits. If you know of a suitable connector, please let me know and I'll update the parts list.

You might prefer to use pin headers and plugs (such as the ones in the parts list in the LexWiz emulation section) to connect to the KL25Z. That would be a little neater than what I did. The main reason I soldered wires directly to the KL25Z is that the sensor requires connections to both the J9 and J10 jumpers, and the calibration button *also* needs connections on both jumpers. I didn't see a way to arrange things using the pin headers so that I'd have independent plugs for the sensor and for the calibration button. So instead of using pluggable connectors directly on the KL25Z, I soldered wires to the KL25Z and ran the wires to a pluggable connector.

When you're done, each terminal of the CCD sensor except for pin 11 should be connected to something – either another pin on the sensor, or a pin on the KL25Z (or both). Pin 11 on the sensor is to be left unconnected.

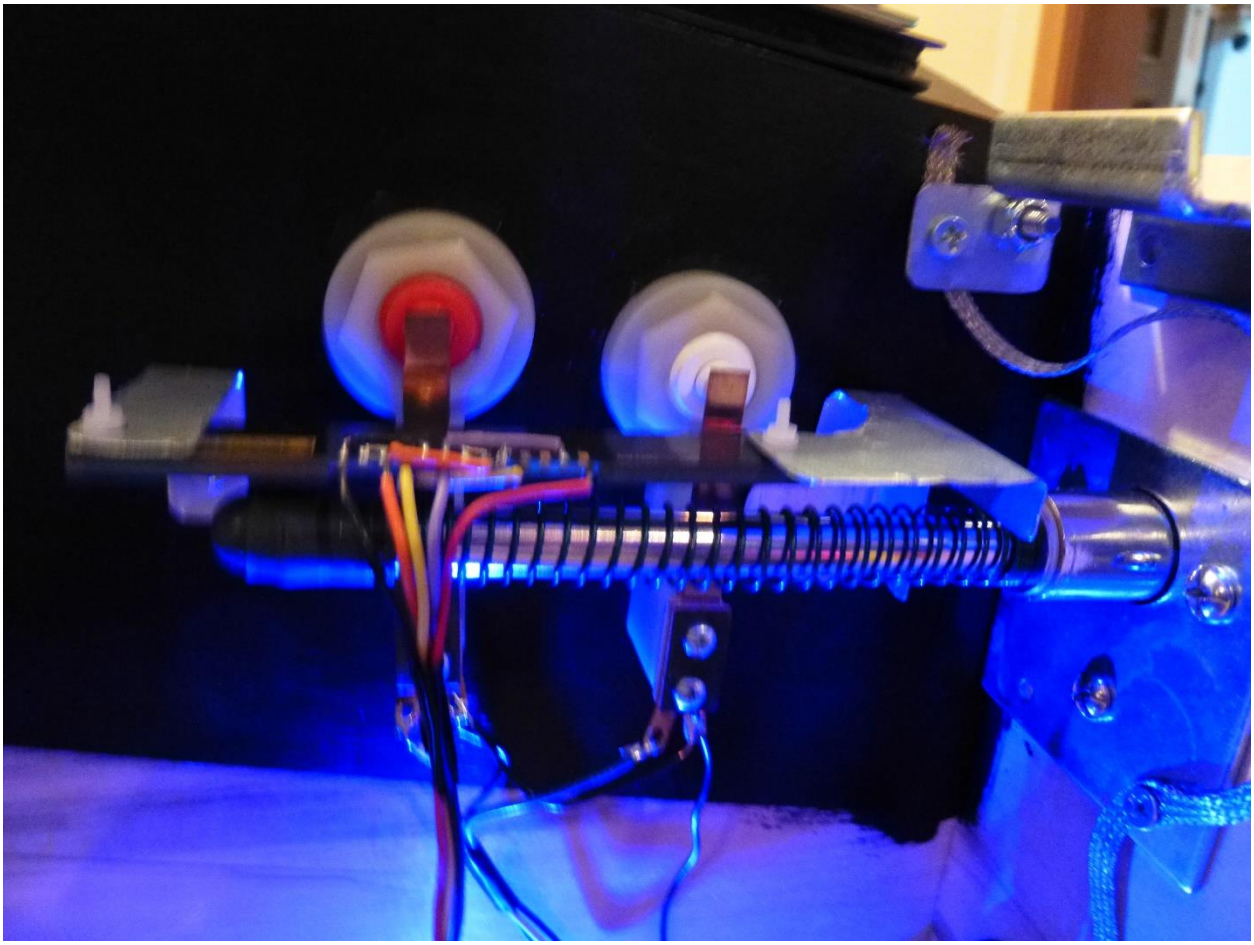


Mount the plunger sensor

This is an area where you'll have to improvise. I don't know of an off-the-shelf bracket that will hold the sensor at the proper position. I made two custom L brackets (one for each end of the sensor) out of sheet metal – I snipped them to size, bent them at a 90° angle, and drilled holes for screws. The sensor

is quite lightweight, so there's no need for anything especially strong, but you'll have to make sure it's solidly held in place so that the alignment doesn't change during use.

Here's what my finished installation looks like. The white connectors are M3 nylon screws.



Proper placement and alignment of the device is important, but it's not too hard. I was able to get mine set up entirely by eyeballing it, and it worked on the first try. The key requirements are:

- The sensor must be located above the plunger, with the window facing down
- It should be as close to the plunger as possible, but far enough away that there's no risk of physical contact
- The window should be as close as possible to the center of the plunger's long axis (the point is for the plunger to cast a shadow on the window)
- The ends of the window should extend slightly past the full range of motion for the plunger

The last point is really the key to the whole scheme, and probably the trickiest to get right. The software needs to see one end of the window fully lit, with no shadow, and the other end in full shadow.

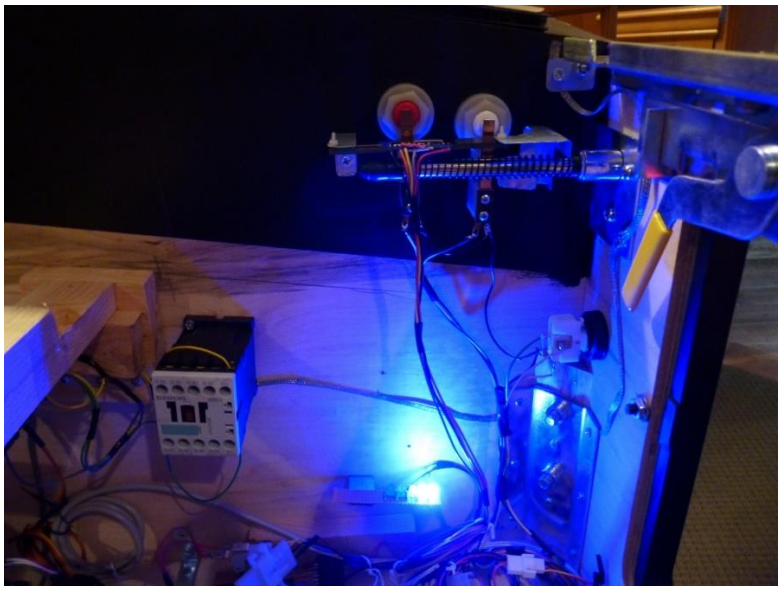
It needs to see both extremes on each reading to determine the relative light levels for the exposure. This is part of what makes the sensor tolerant of a range of lighting conditions. When the plunger is pushed all the way forward, at least a few pixels have to be left uncovered by the shadow, so you need to make sure the sensor extends a little bit beyond the fully forward position. Likewise, a few pixels have to remain in shadow when the plunger is pulled all the way back. The sensor is about a quarter inch longer overall than the full travel range of a standard pinball plunger, so you have about 1/8" to work with at each end.

Note that the plunger can travel about half an inch forward from its rest position if you push it and compress the barrel spring on the front. It will do this naturally when you fire it with full force, so it's important to take this into account for the sensor positioning.

It doesn't matter whether the sensor is mounted with the contacts at the left or right, as long as the window faces down. The software figures out which direction is which from the light pattern when it reads the pixels. It knows the brighter end is the tip.

Install the plunger light source

In my testing, the CCD seems tolerant of a pretty wide range of light conditions, particularly on the low end. Too much light will overexpose it and prevent the software from detecting the shadow edge, but it seems almost impossible to have too little light. In my tests, the sensor worked (although not with perfect reliability) from nothing more than the ambient light inside my cabinet from various indicator LEDs on the motherboard and other devices. It seems most reliable with a small dedicated light source positioned about 8" away. The arrangement that's working very reliably for me consists of two 20mA blue LEDs mounted on the side wall of the cabinet, near the floor, about 8" away from the sensor. They're positioned roughly in line with the front of the sensor.



The principle of operation of the plunger sensor is to detect the edge of the shadow cast by the plunger against the sensor. We therefore want as sharp a shadow as possible. In my testing, it appears that the

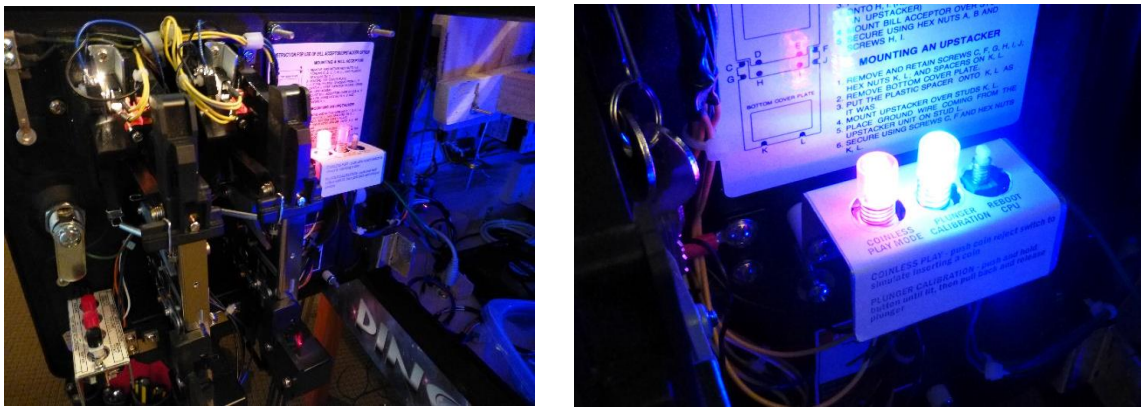
shadow is sharper when the light source is further away. This makes sense from an optics perspective: a distant light source will look more like a point source, so the angular distribution of the incoming rays will be narrower.

Install the plunger calibration button

This step is optional. If you don't install a button, you can use the configuration tool to activate calibration mode from Windows. That's not quite as spiffy, but it saves some work.

On my cabinet, I mounted my plunger calibration button on the inside of the coin door, in a little cluster of my own special-purpose service buttons. My extra buttons are, first, a lighted toggle button that enables and disables "virtual coin mode" (when enabled, you can push one of the Coin Reject buttons on the coin door to simulate dropping a quarter in the slot); second, the plunger calibration button; and third, a PC Reset button wired to my PC motherboard, to do a hard reset on the PC if Windows should ever freeze. (I haven't had to use this yet. Hooray for Windows 7.)

I found some pushbuttons with built-in indicator LEDs that resemble the ones Williams used for their coin door service buttons. My "virtual coin mode" button lights up red when switched on, and my plunger calibration button lights up blue to provide status feedback. I mounted the buttons in a little bracket I fabricated from sheet metal; it looks pretty close to the real thing, and the functionality has been just what I wanted.



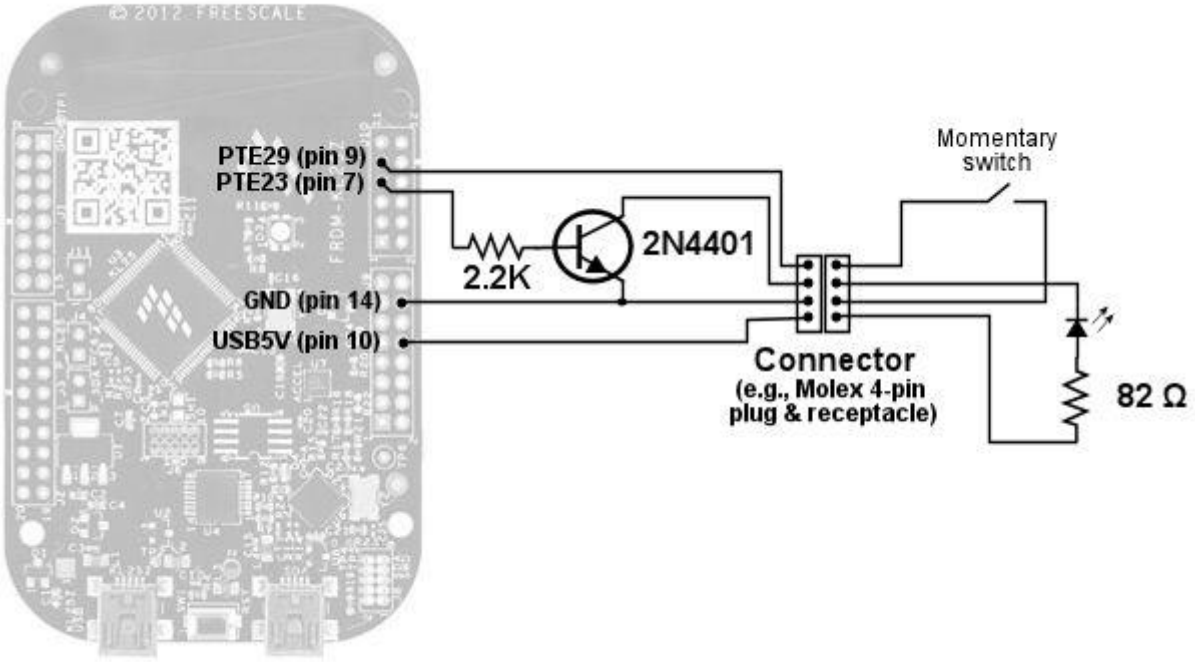
Left: the inside of the coin door. My added service buttons are in the bright cluster in the upper right. This space is normally reserved for a dollar bill acceptor, which I have no need for. Right: a close-up of my custom buttons. The middle button, glowing blue, is the plunger calibration button.

If you do install a calibration button, I recommend placing it somewhere inside the machine, or perhaps on the bottom, so that curious guests aren't constantly pushing it to see what it does.

Hooking up a plain button without an indicator LED is easy. Just wire the terminals of any momentary switch between the KLZ25's PTE29 (jumper J10 pin 9) and ground (jumper J9 pin 14).

If you use a plain button, you can still get the same feedback that the indicator LED would provide by observing the diagnostic LED on the KL25Z. It will display the same flash pattern that the button indicator would when you press and hold the button.

If you want the fully decked out version, order the illuminated button from my parts list and construct the circuit shown below. The transistor's purpose is to boost the 4mA output current that the KL25Z pins can provide to the 20mA level needed to power the LED. I built this part of the circuit on a tiny little 1x1 cm perf board and just made it part of the bundle of wires between the switch and the KL25Z.



On the KL25Z side, I soldered the four wires directly to the solder pads for the jumper pins, and attached the other ends of these wires to the Molex 4-pin plug as shown in the schematic. I likewise soldered wires to the switch terminals and ran those to the matching 4-pin socket. You might prefer to install vertical pin headers on the KL25Z solder pads instead of wiring directly to the pads. (For sample parts, see the parts list for the LedWiz output section, which uses the type of headers I'm talking about.) The reason I wired directly to the KL25Z rather than using headers is that the arrangement of pins on these particular KL25Z headers made it impossible to arrange things so that the switch could have its own connector independent of the CCD connector. The CCD and switch each need to connect to a mix of pins on J9 and J10. If I could have had one device connect entirely to J9 and the other to J10, I would have used pin headers, but as it is I would have had to interpose another plug and socket to make the two bundles independent. I didn't want two sets of connectors, so I just soldered directly to the KL25Z. You can still remove the KL25Z from the machine without having to remove any other parts, since everything still connects with plugs; you just have some wires dangling from it when you do.

In the schematic above, I put the transistor circuit on the KL25Z side of the plugs, and the LED resistor on the switch side. This was intentional. The transistor circuit will work with any indicator light (up to about 600mA), but the 82Ω resistor is sized for the particular LED in the switch in my parts list. If you

should want to change to a different switch or indicator, the resistor value will probably have to change with it, so I grouped it physically with that part of the circuit. That makes the indicator switch more modular. Electrically, of course, it makes no difference which side of the connector it's on.

Initial testing and troubleshooting

The configuration tool (described earlier) lets you view the raw light-level readings from the CCD sensor. Once you have the sensor wired and installed, make sure that the KL25Z is plugged in to your PC via USB, then fire up the configuration tool in Windows. The tool should find your controller and show it in the drop-down list. Click on the "View CCD Exposure" button to bring up the pixel display. This shows a graphical depiction of the sensor pixel readings, with the brightness at each pixel shown in grayscale. Black means that the pixel isn't seeing any light, white means that the pixel is fully saturated, and shades of gray show intermediate brightness levels.

The pixel display is updated continuously, so if it looks right so far, you can try pulling the plunger. The on-screen display should track plunger movement – you should see the dark area shrink as you pull back the plunger. On the other hand, if the initial display looks too bright or too dark, you can try adjusting the light source, and you should see the results immediately reflected on-screen.

Ideally, with the plunger at the rest position, you should see a short stretch of white (or nearly white) pixels, representing the area beyond the tip of the plunger where there's no shadow, and the rest black (or nearly black), representing the area in the shadow of the plunger rod. There should be a nice sharp edge between the two. If the pixels are all black, the sensor is underexposed – you need more light. If the pixels are all white, or nearly so, it's overexposed – you need to reduce the brightness of the light source or move it further away from the sensor. If the pixels look random, it might indicate a wiring problem.

In my testing, the sensor has proved to be very tolerant of low light, but can easily get oversaturated by too much light. Make sure that the top side of the sensor isn't getting hit by any ambient light from outside the cabinet. This shouldn't be a problem if your machine is assembled and closed up, but if you had to take it apart for testing, put something opaque on top of the sensor to block ambient light. Try adjusting the distance and brightness of your light source. Dimmer is better. If you're using something brighter than a couple of small LEDs, try partially covering the light source to make it dimmer.

If you can't get any response by adjusting the light source, double-check your wiring, to make sure everything's connected to the proper terminals as shown in the schematic. You might also want to check that you didn't break any solder joints in the installation process.

If the raw pixel status looks good, the next thing to check is that everything is working at the USB joystick level. Bring up the Windows control panel called "Set up USB game controllers" and open the Pinscape Controller. The X/Y position on the joystick display should respond to cabinet nudges – that's the accelerometer. When you move the plunger, you should see the Z axis change. When the plunger is at rest, the Z axis position on the display should be approximately at the center (zero), and when it's pulled all the way back, the axis should be close to the maximum in the positive direction.

Calibrate the plunger

The plunger sensor needs to be calibrated when first installed. This lets the software measure precisely which pixels are within the range of the sensor, so that it can report an accurate zero point and an accurate maximum retraction point to Visual Pinball.

The calibration results are stored in non-volatile flash memory on the KL25Z, so you don't have to repeat this process when you turn off the computer or disconnect the KL25Z. The flash memory record survives power cycles and disconnections. However, you will have to recalibrate any time you reinstall the KL25Z software (including upgrades), because installing new software erases the whole flash memory. You should also recalibrate if you ever have to remove the sensor or the mechanical plunger itself – the relative positions will probably shift slightly when you put everything back together, so it's a good idea to recalibrate to make sure that the zero point is exactly right again.

The calibration procedure takes about 20 seconds. If you installed a calibration button, push it and hold it. It will flash (as will the diagnostic LED on the KL25Z) for about two seconds, then turn solid on. (The delay is simply to avoid triggering calibration if you briefly push the button accidentally.) Solid on means that calibration is in progress. This lasts about 15 seconds. Simply draw the plunger all the way back, hold it for a few seconds, and then slowly return it to the rest position. Don't let it spring back from the retracted position, and don't push it forward beyond the normal rest position. When the 15 seconds is up, the button light will turn off (the KL25Z LED will return to its usual flashing pattern).

If you didn't install a calibration button, run the configuration tool on Windows and select the plunger calibration mode option. The KL25Z LED will immediately turn solid blue to indicate that calibration is in progress. Go through the same motions with the plunger as above.

During calibration, the software notes the endpoints of the plunger travel. It uses the farthest forward position as the zero point, and uses the maximum retraction point to determine the overall range. This allows it to report positions that accurately align with the corresponding reference points on the simulated plunger in Visual Pinball.

Modify your Virtual Pinball tables to work with mechanical plungers

Some of the Virtual Pinball tables in circulation support a mechanical plunger out of the box. Most don't. Fortunately, support can be added to most tables with a modest amount of work. I haven't reduced this to a science, and there's enough variation in table design that I don't think it's possible to do so. You'll have to adapt the generic procedure to each table, and the generic procedure doesn't work at all on some tables. And, of course, you'll have to repeat this procedure for each table.

Here's an outline of my approach. Visual Pinball has a built-in plunger object class that handles the mechanical plunger. When this object is used to implement the on-screen plunger, the table usually works right out of the box. Most tables in circulation don't do this, though. They instead use scripting and custom objects to implement the on-screen plunger. The scripts don't usually handle joystick input, so the mechanical plunger won't work. However, these tables usually *also* have one of the built-in plunger objects, which they use as a helper for the scripts. They usually either hide this object or move it off-screen so that it doesn't provide the visuals and can't come into contact with the ball in the

simulation. My approach to "fixing" these tables is to find the built-in plunger object and move it into position so that it can strike the ball and launch it. The built-in plunger object can usually be left hidden so that the custom visuals are still used, but we do want it to provide the launch physics directly.

To start working on a table, open it in the Visual Pinball editor.

Find and select the built-in plunger object, if there is one. The easiest way to do this on complex tables is to use the Select Element command (Ctrl+Shift+E), and find the object called Plunger in the list. If the table doesn't have an object called Plunger, look for something with a similar name, like Plunger1. If you can't find any obvious candidate, this table might one of the (hopefully) rare cases that requires work beyond this recipe.

You should see a highlighted rectangle showing where the plunger is located in the table layout. If the plunger is positioned anywhere but the standard spot, you're dealing with a table where this object is hidden and only exists for scripting purposes. The table is designed so that this object never touches the ball. Even so, the next thing I do is to move it into the standard position so that it *does* strike the ball when released. This contradicts the table author's intention, and might break their plunger scripts. But remember, their scripts don't work with our mechanical plunger, so we're never going to trigger them anyway.

Next, we have to check some property settings on the plunger object. If the Options panel isn't already displayed on the right side of the window, click the Options button in the left tool palette to bring it up.

In the Options panel, check the "Enable Mechanical Plunger" box. This controls the built-in joystick handling – when disabled, VP ignores the joystick input for this object, so we need to make sure it's enabled. Also set the Type to "PlungerTypeModern", if it's not already, in case we end up using it for the visuals. (The "Modern" type looks like a real plunger; the "Orig" type is cartoonish.) Finally, the Park Position should be set to 0.16667.

We're now ready to give it a try. Press F5 to run the table.

If the plunger on-screen now tracks the movement of your real plunger, start a game and try launching a ball. On some tables this will work at this point, in which case you're done. On others the plunger will move but it won't have any effect on the ball.

The next step, if the plunger doesn't strike the ball, is to try moving the plunger object upward a bit. You can either move it by dragging the layout object, or type a new Y value into the Options panel. Try moving it up by about 40 as a first guess, then run again and see if it makes any difference. You might have to try several times to zero in on the right position.

If you can get the ball to react to the plunger, but the launch speed is too slow or too fast, adjust the Mech Strength value in the plunger object properties. Higher strength makes for a faster launch.

If you get the launch action working, but the visuals show two plunger objects, one on top of the other, the table is using scripting to display its own custom visual for the plunger. When we moved the built-in plunger object into the same position, we created the double visual. The custom visual is usually the more authentic looking of the two, but not always. If you like the custom visual better, select the built-

in plunger object again and un-check the Visible box in its properties. It will still be "physically" present in the simulation, but won't be displayed. The custom visual is often not animated as smoothly as the real plunger, though, so you might prefer to hide it and use the built-in visuals. In this case, you'll need to identify the custom object (which isn't always easy) and un-check its Visible box.

Wire the cabinet button inputs

You can optionally use the controller to wire your cabinet buttons (flipper buttons, Start button, etc.) to the PC. If you do, they'll appear to the PC as joystick buttons. Visual Pinball can read up to 32 joystick buttons, and it lets you assign a function to each button using its Keys dialog.

To wire a button, simply connect one of the button's switch terminals to the KL25Z Ground (jumper J9 pin 12 or 14), and connect the other switch terminal to the pin shown in the chart below for the selected button number.

Note that every button has one terminal connected to the KL25Z Ground pin, so it's fine to daisy-chain the ground wiring from one button to the next. That is, rather than running a ground wire all the way from a button to the KL25Z Ground pin, you can instead run the wire to another nearby button's Ground terminal. This can save a lot of wire, since your cabinet will probably have a few clusters of buttons. Each button's other terminal must be wired to a separate KL25Z input port, though, so you will need to run one wire from each button all the way to the board.

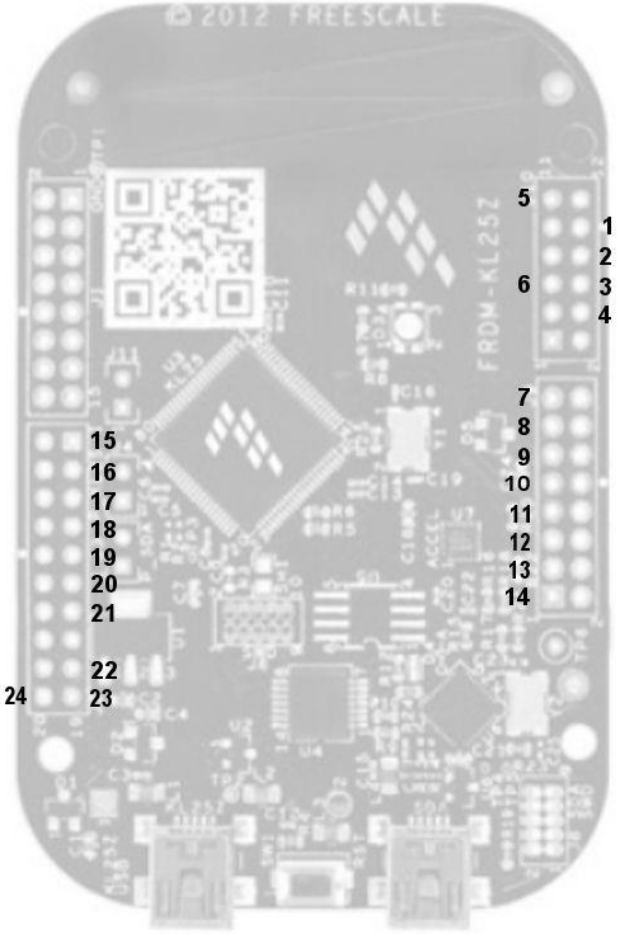
To test your button wiring, you can use the "Set up USB Game Controllers" control panel in Windows. Double-click the Pinscape Controller entry. This will display the status of the 32 buttons. When you push a physical button on your cabinet, the corresponding joystick button should light up on the control panel display.

Once you've physically wired your buttons, you'll need to configure them in Visual Pinball. To do this, open VP without any table loaded, and use the Preferences > Keys menu to open the keyboard dialog. Each function (Left Flipper, Right Flipper, Start, etc) has a drop-down list underneath. For each button, find the function that you want to assign to the button, click on its drop-down list, and select the button number corresponding to the KL25Z pin that you wired the button to.

Here are the default button pin assignments. You can change these, but to do so you have to edit the source code for the controller software. You're free to do that since it's an open-source project. Note that the USB report we use can handle up to 32 buttons, but we've only assigned 24 by default, since the KL25Z has a limited number of pins available. If you're not using the LedWiz features, you can reassign ports currently used for the LedWiz emulation as button input ports. You can find instructions for reassign pins as inputs in comments in the main source code file (main.cpp) in the mbed.org repository. Look for the **buttonMap[]** array definition.

Button Input Pin Assignments

Button Number	Port Name	Jumper/ Pin No.
1	PTC2	J10 pin 10
2	PTB3	J10 pin 8
3	PTB2	J10 pin 6
4	PTB1	J10 pin 4
5	PTE30	J10 pin 11
6	PTE22	J10 pin 5
7	PTE5	J9 pin 15
8	PTE4	J9 pin 13
9	PEE3	J9 pin 11
10	PTE2	J9 pin 9
11	PTB11	J9 pin 7
12	PTB10	J9 pin 5
13	PTB9	J9 pin 3
14	PTB8	J9 pin 1
15	PTC12	J2 pin 1
16	PTC13	J2 pin 3
17	PTC16	J2 pin 5
18	PTC17	J2 pin 7
19	PTA16	J2 pin 9
20	PTA17	J2 pin 11
21	PTE31	J2 pin 13
22	PTD6	J2 pin 17
23	PTD7	J2 pin 19
24	PTE1	J2 pin 20

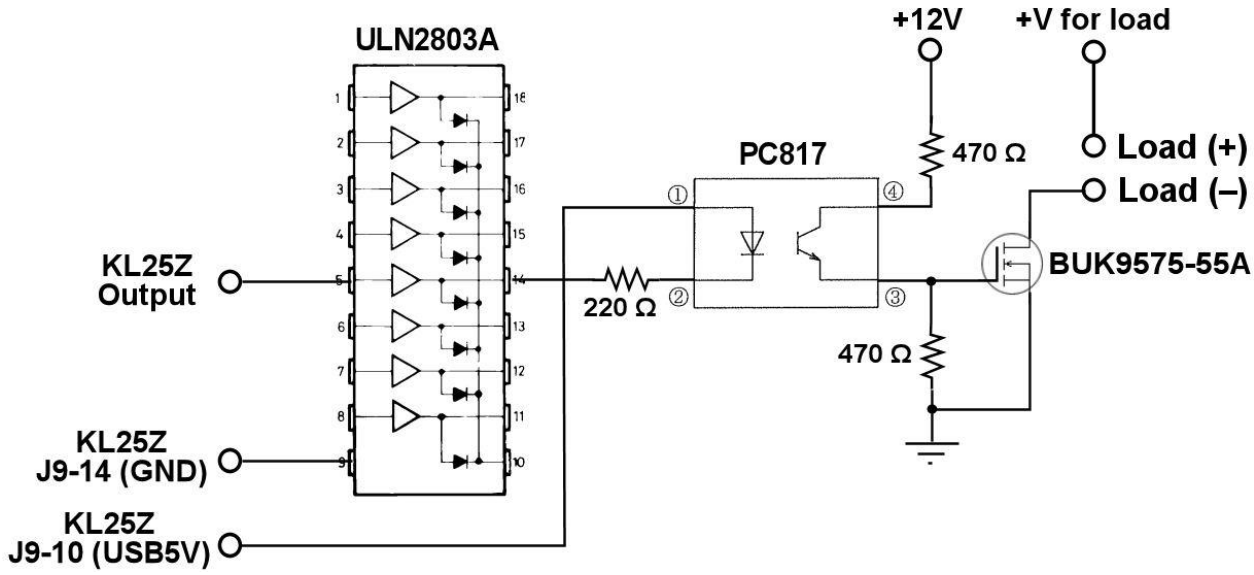


Build the LedWiz output drivers

If you bought the parts in my shopping list for the LedWiz emulator output stage, you'll be able to build an output driver for each port that will handle just about any device you can throw at it, without having to worry about using relays to control high currents. The MOSFET I spec'd in my shopping list will handle up to 20 Amps and voltages up to 55V. This will easily handle enough current for a shaker motor, replay knocker, contactors, or even a flipper coil. It's also fast enough to pass through PWM signals, so it can control the brightness on flasher LEDs or the speed of a shaker motor. The high current and voltage is isolated from the KL25Z (and your PC motherboard) with an optocoupler, so there's very little chance of any voltage surges making it through this firewall should anything break on the high-power side. (Even so, you should put a fuse in-line with each feedback device to protect the MOSFET and your power supply from short circuits or overloads in the feedback device.)

One big caveat: don't hook up anything involving AC power to these circuits. MOSFET drivers are strictly for DC. If you want to attach an AC device, put a relay on the output, and switch the AC with the relay.

Here's the schematic for the circuit you need to build for each output. I recommend building this on a perf board so that all the components are secured in place. Virtuabotix has a nice selection of these in various sizes at reasonable prices. WARNING: This circuit is preliminary and not fully tested.



Note that the input and output pins shown on the ULN2803A chip are for illustration only. This chip has 8 identical blocks, arranged with their inputs and outputs on the corresponding pins on opposite sides of the chip. You can therefore use one ULN2803A to build 8 copies of the circuit shown above. Just wire each one to a separate pin pair on the ULN2803A.

Here's how the circuit works. The ULN2803A amplifies the signal from the KL25Z output pin to the level needed for the PC817 optocoupler. The optocoupler in turn switches the BUK9575-55A MOSFET on and off. The MOSFET provides the actual load switching.

The "+V for load" is the positive supply voltage appropriate for your load. This can be up to 55V, assuming you use the BUK9575-55A, and the current limit is 20A. Note that you should always connect a diode (a 1N4007 works for most purposes) across the load terminals for any sort of inductive device (a motor, relay, contactor, solenoid, etc.). Connect the diode with the polarity reversed: the "bar" on the diode should connect to the positive (+V) terminal on the load.

Note that you can make do with just the ULN2803A for very small loads, like 20mA LEDs. The ULN2083A can nominally handle 500 mA per output, so it's safe to drive a small load directly. However, for anything bigger, like a contactor or motor, you should use the whole circuit. That will provide plenty of headroom for all of the common virtual pinball feedback devices.

There are other ways to design a power switching circuit. Virtual pinball builders often use relays because they're simple and can handle tons of current. I prefer a MOSFET driver because it's about as cheap and almost as simple to wire as a relay, and switches so much faster (fast enough for PWM). Note that if you do want to use a relay (to switch AC current, for example), you can probably drive it directly from the UL2803A – just make sure the coil actuation current is well below 500mA.

Pin assignments for LedWiz emulation output ports

The charts below show how the output pins on the KL25Z are assigned to software port numbers in the LedWiz emulation.

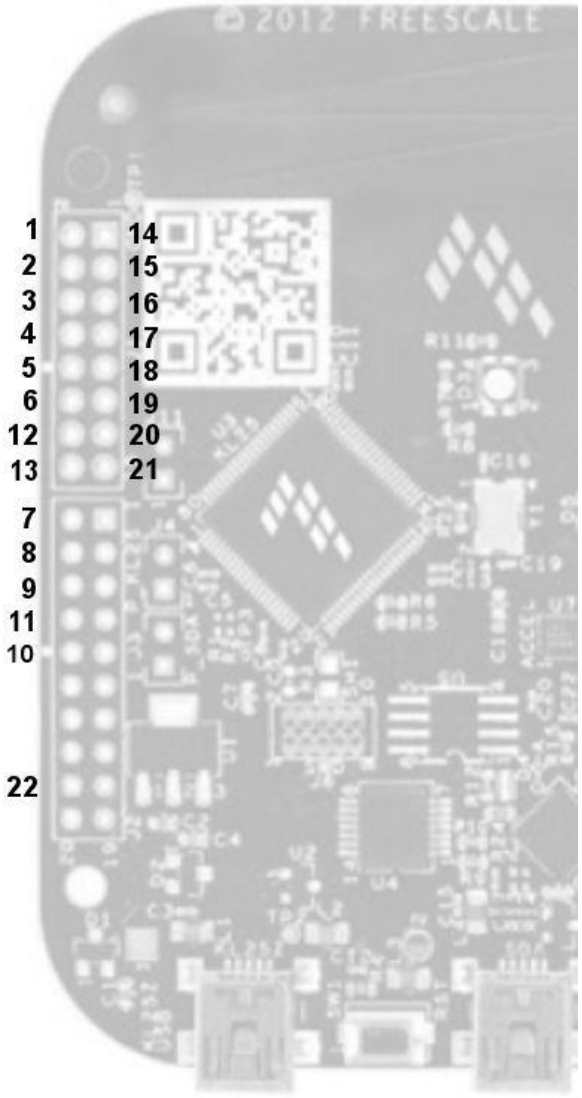
You'll notice that the LedWiz port numbers aren't arranged in simple sequential order on the physical jumper pins. I would have liked to arrange things that way, but it wasn't possible given a more important constraint: I wanted to keep the PWM-capable ports grouped together in a single block in the LedWiz numbering scheme. PWM ports are especially useful for devices like LEDs that can benefit from brightness control, and a special case of LEDs is the RGB LED. A single RGB LED looks like three separate devices to the LedWiz, because each color needs its own independent output. For easiest software configuration on the PC side, the three color channels for a given RGB LED need to be grouped together into a sequential block of port numbers. The KL25Z only allows certain pins to be assigned as PWM outputs, so in order to keep the PWM ports together in the LedWiz numbering, I had to scatter the LedWiz numbering around a little bit to work within the KL25Z limits.

In the port assignments below, **Ports 1-10 are PWM-capable.** Other ports can only be turned fully on and fully off.

You'll also notice that some of the LedWiz port numbers aren't assigned to physical pins. This is because the KL25Z has a limited number of pins available, so we have to budget them among the various features (plunger, LedWiz emulation, and key inputs). We figured that no one will try to use this controller as a full LedWiz replacement, since it's not really capable of that anyway (see the section on LedWiz limitations below), so a few missing LedWiz ports won't bother anyone. If you really do want the full 32 output ports, you can modify the source code to add the missing ports. Just about all of the physical pins on the board are spoken for in the default configuration, so you'll have to reassign pins that are currently used for other features that you're not using, such as button inputs. You can find instructions for reassigning pins as LedWiz outputs in comments in the main source code file (main.cpp) in the mbed.org repository. Look for the **ledWizPortMap[]** array definition.

LedWiz Pin Assignments

LedWiz Port No.	KL25Z Jumper-Pin	KL25Z Pin Name
1	J1-2	PTA1
2	J1-4	PTA2
3	J1-6	PTD4
4	J1-8	PTA12
5	J1-10	PTA4
6	J1-12	PTA5
7	J2-2	PTA13
8	J2-4	PTD5
9	J2-6	PTD0
10	J2-10	PTD3
11	J2-8	PTD2
12	J1-14	PTC8
13	J1-16	PTC9
14	J1-1	PTC7
15	J1-3	PTC0
16	J1-5	PTC3
17	J1-7	PTC4
18	J1-9	PTC5
19	J1-11	PTC6
20	J1-13	PTC10
21	J1-15	PTC11
22	J2-18	PTE0
23		
24	-	-
25	-	-
26	-	-
27	-	-
28	-	-
29	-	-
30	-	-
31	-	-
32	-	-



Limitations of the LedWiz emulation

I want to return to a point I raised in the introduction: the Pinscape Controller's LedWiz emulation has some limitations. The LedWiz emulation is really just a bonus feature anyway, so I don't expect anyone will run up against these limits in practice, but here are the details:

- PWM is only available on 10 ports. (PWM is pulse-width modulation, which is used to control lamp brightness and motor speeds.) This is a hardware limitation; the KL25Z hardware only provides 10 PWM channels. I mapped the virtual LedWiz ports 1 through 10 as the PWM outputs. (You can rearrange the mapping of ports to pins, if you're willing to modify the source

code. Even if you do, you still can't have more than 10 PWM ports.)

The controller accepts commands to set the intensity level on all ports, but it ignores level settings on the non-PWM ports. The non-PWM ports can still be turned on and off, of course, so they're suitable for devices like contactors that have no need for intensity settings. If you want to connect a shaker motor or flasher LED, use one of the PWM ports for that device.

- The Pinscape Controller doesn't support any of the LedWiz flashing light modes. These modes allow the host to set a flashing pattern on a light without having to continually send commands to control the flashing. They're used by some host software, such as Led-Blinky. However, I don't think DirectOutput Framework uses these at all, so if you're using DOF (which you should be), this limitation shouldn't matter. It also won't matter for devices like contactors and knockers, where you'd never want to use a flashing mode in the first place.

Future work

Here are some ideas for future enhancements to the project.

• End-of-stroke switch for plunger: The release motion of the plunger is much too fast to capture in the CCD. We can read a frame in about 25-30ms; the plunger release motion takes less than that. (I haven't been able to measure it exactly because I don't have any equipment that can do photography fast enough.) When you release the plunger without having a ball for it to strike, it shoots all the way forward and bounces back off of the barrel spring. The rebound brings it back to some fraction of the original pull position. This process repeats 8-10 times over 300-500ms until the energy damps out and the plunger comes to rest.

Each bounce cycle takes less than one CCD frame, which makes things a bit tricky for the software. The release motion is so fast that we can't capture it frame by frame. The plunger can make it all the way home and bounce back almost (but not quite) to where it started by the time we can read the next frame. This means that during a release, the position that we observe on each frame is *almost* random – it's somewhere between the start (of the pull) and the home position, but exactly where depends on the exact timing of when we happen to take the frame. We could catch it all the way at the home position or at the apex of the current bounce, or anywhere in between.

Fortunately, though, there is a pattern we can exploit. The bounces follow an exponential decay envelope: each bounce's apex will be a fraction of the last apex. This means that we'll reliably observe a series of positions that are all below the starting position. Not monotonically, though – if we happen to catch frame A at the bottom of a bounce and frame B at the top of the next bounce (or the next one after that), frame A and B will appear to be out of order. But both A and B will definitely be below the original pull position, and as we observe frames C and D and E, we'll see an overall declining trend.

The software uses this to detect a release. It assumes that any time it sees a steady reading at a retracted position, followed by a couple of frames well below that position, a release has started. It sets a flag internally when it detects this, reports a zero position to VP, and then waits until the plunger comes to rest before reporting the actual position again. This works pretty well, but the user can trick it by doing a fast but controlled forward motion that doesn't actually reach the home position. The

software will think it's observing a release when it sees the start of the motion, so it'll report this to VP, and VP will launch the ball.

It's probably not necessary to fix this, because I don't think most users would do this naturally – they'd only do it if they were *trying* to trick us, in which case it's okay to be tricked. But if further experience shows that it's a practical problem after all, I think I have a solution. What we need is a much faster sensor that can tell us with certainly that a release has occurred, rather than trying to infer this from the slow (relative to the plunger motion) CCD sensor. We'd still use the CCD sensor just like we do now – I still think it's an ideal solution for precise position sensing with the plunger. But we'd supplement the slow-but-precise position readings with a second, coarse-but-speedy sensor that tells us instantly when the plunger hits the home position.

The sensor I have in mind is extremely simple: a mechanical switch. Some people already use end-of-stroke switches as simple plunger inputs on their cabinets, encoding it as the Enter key to trigger VP's timed plunger pull scheme. In our case, we'd position the switch so that the plunger hits it at the end of the stroke – when at the home/rest position. Pull it back just slightly and the switch toggles. We'd connect this switch to a KL25Z GPIO pin that we configure to generate an interrupt. The interrupt handler sets a flag whenever the switch transitions from open (plunger not at home) to closed (plunger at home). At the end of each CCD read cycle, we check the flag. If the flag is set, we know that the CCD reading is already out of date, because we have more timely information from the switch that the plunger is actually at the home position. At this point, we simply trigger the exact same code we have now for a release. Everything else works the same way it does now; we just get a more bulletproof indication of when a release occurs.