The Pinscape Controller

A How-To Guide

M. J. Roberts • August, 2014

Revised August, 2015

Pinscape is the name of my virtual pinball cabinet, which I've been building for a little over a year now. The Pinscape Controller is a device I designed as part of that project. Its original purpose was to handle two special types of virtual pinball input, namely sensing the plunger position and sensing accelerations from nudging the cabinet, but the project expanded to handle button input wiring and feedback device control as well. 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 with its built-in joystick driver. It also works seamlessly with Visual Pinball, which has built-in support for joystick input for nudging and plunger position sensing.

Physically, the main controller runs on a KL25Z microcontroller board. This is a credit-card sized, self-contained, single-board computer that costs about $13. 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. This is packaged as a small circuit board, about 3" x 1/2". You mount this adjacent to the mechanical plunger, and run some wires between it and the KL25Z. You'll also need to install a small light source to illuminate the sensor. The CCD is essentially a camera; it measures the plunger position by rapidly taking pictures of the plunger, which the software analyzes to determine how far the plunger is pulled back at any given instant.

Alternatively, the software can sense the plunger position using a slide potentiometer, which is cheaper and somewhat simpler than the CCD. To set this up, you mechanically attach the plunger rod to the pot's slide lever, so that moving the plunger moves the lever and changes the pot's electrical resistance. Since plunger position and resistance vary in sync, the software can figure the position from the resistance, which it can measure via the microcontroller's analog voltage sampler.

Features

The controller has four main functions. These features work independently; you can use any subset.

- Plunger position sensing. The software works with a CCD sensor or potentiometer 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. This is a huge enhancement to realism and immersion that you can't get in desktop virtual pinball.

- 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. Wiring buttons is fairly simple; you just wire each button to a pin on the KL25Z headers.

- LedWiz-compatible output control for feedback devices. This feature has some significant limitations, so you shouldn't think of the KL25Z as a full replacement for a real LedWiz, but it comes in handy if you need a few extra ports. Note that some fairly involved soldering work is required to use this feature.

The competition

There are (at least) three commercial plunger-and-nudge solutions available: Zeb's Digital Plunger Kit, the VirtuaPin Plunger Kit, and the Nano-Tech Mot-Ion Controller. Zeb's kit uses a linear encoder for the plunger sensor; the other two use IR proximity sensors. All three kits also provide accelerometer nudge sensors and button input encoding. They're all plug-and-play kits that work right out of the box, and they come with warranties and customer support. I'd strongly recommend looking at these options before considering Pinscape controller, unless you really enjoy challenging DIY projects.

The big reason I didn't use one of those kits myself is that I wasn't satisfied with the precision or stability of the position sensing of the IR-based kits. (Zeb's kit wasn't on the market at the time, so IR was the only option.) I want to emphasize that I'm pickier than most people about this. The IR sensors do work and do perform the basic job, and most people seem reasonably happy with them. But in my testing, they weren't as precise as I wanted. I'm particularly fond of some tables with tricky plunger skill shots, which can't be realistically simulated without fairly high precision from the plunger sensor.

Note that Zeb's kit didn't exist when I started this project. It uses a different sort of sensor that looks like it achieves the precision I was aiming for. I haven't seen it in action personally, but reviews on the Web forums have been quite positive. I might not have even started on this project if Zeb's kit had been available at the time.

No Warranty

Just to be sure there's no misunderstanding, I want to make clear that this whole project has NO WARRANTY of any kind. I'm making all of this 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 make the documentation as complete and accurate as I can, but I might have missed important details or made errors. Likewise, the software might (and probably does) contain bugs. 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

You'll need two main suppliers for parts for this project, and for your virtual pinball cabinet build in general. The first is an electronics distributor. I've been very happy with Mouser Electronics, and DigiKey also has a good reputation. The second is a pinball parts dealer. Pinball Life and Marco Specialties are both great. I also recommend VirtuaPin; their prices are usually quite good, and they have many of the key parts for virtual cabinets, but their selection isn't as comprehensive as the distributors'. eBay can also be worth a look, although used pinball parts on eBay always seem to be at least as expensive as new parts from the distributors, and aren't as shiny and new.

The specific part numbers and supplier links below are only suggestions, to make shopping easier. Feel free to substitute similar parts as you see fit.

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 CCD (optical sensor) 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 slide potentiometer alternative plunger sensor:

- ALPS RSA0N11S9A0K, *or* Bourns PTF01-152A-103B2, *or* this part, *or* any slide pot with linear taper and 80mm+ travel

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 cable length based on 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. The KL25Z actually has *two* USB ports, one for application use (in our case, the joystick interface) and the other for firmware programming. If you want to keep both ports plugged in all the time for easy firmware updates in the future, buy two USB cables.

** 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 plunger will never actually strike a ball, so it doesn't need a lot of force; excess force will just make it bounce more when it hits the barrel spring. I think it also 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 some basic tools:

Soldering iron (or, better yet, a soldering station)
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" to update boot loader – it should have a name like BOOTUPDATEAPP_Pemicro_vxxx.SDA
- Use the same procedure to install MSD-DEBUG-FRDM-KL25Z_Pemicro_vxxx.SDA (from the same zip file)

Attention Windows 8 users! Some versions of the factory-installed boot loader **cannot be used or updated** on a Windows 8 machine. If you're using Windows 8, and the update process above doesn't seem to work, try again on a machine running an earlier Windows version (XP, Vista, or Win 7). Once you get the updated boot loader installed, everything will work fine on Windows 8, but you won't be able to use Windows 8 to do that first boot loader update if the buggy version was installed on your KL25Z at the factory.

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.

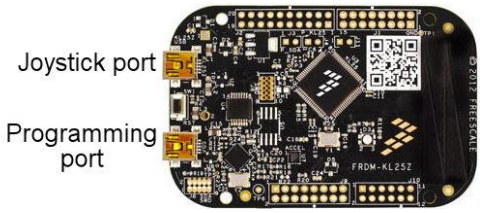
Customize the controller software (config.h)

Before you install the software, you might want to take a look at the customization options. This is an open-source project, so you're free to customize the entire body of the source code as much (or little) as you like. Even if you don't know how to code in C++, there are a number of options settings that you can choose with a little simple text editing. The options are centralized in a single source file with all of the configuration settings, called config.h.

To edit this file and build a customized version of the firmware, follow these steps. On the mbed project page, click on the button "Import this program". You'll need to create a free mbed account to use this feature, because this step will create a private copy of the project in your own work space on the site. Once you create your account and import the program, open the Pinscape Controller folder in the source tree and click on config.h. This file is about 95% comments with a few bits of C++ code that contain the various option settings. The comments explain the available options. Read through the file and see if there's anything you want to change. Make your edits, save the file, and click the Compile button to create your customized version of the installable binary. Note that you can always return here and make additional changes at any time – you'll just need to download the binary again if you do.

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. The KL25Z will automatically start running the Pinscape software **immediately**. There's no need to unplug it or even reset it – the boot loader will take care of launching the new software as soon as you drop it onto the board.

Note: **don't** use the boot loader update procedure from earlier here. In particular, **don't** hold the reset button while plugging in the card. That procedure prepares the board for loading a new boot loader, which you only need to do once when you first purchase the board.

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". The RGB LED on the KL25Z should start blinking yellow and green at about one-second intervals to tell you it's running.

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.

> **Troubleshooting tip:** if the software **isn't** running correctly at this point (the LED isn't flashing yellow/green, and there's no sign of the new USB joystick device in Windows), you **might** be experiencing a problem with Windows Explorer drag-and-drop that some people have encountered with the KL25Z. On some machines, for reasons no one seems to understand, Explorer drag-and-drop doesn't work correctly with the KL25Z boot loader. Fortunately, there's an easy workaround. Open a CMD command prompt window and use a manual COPY command to copy the .bin file to the KL25Z drive letter.

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. I personally leave both ports plugged in all the time because it makes it easier to update the firmware, which I do a lot since I'm still actively working on it. But there's also no need to leave the programming port plugged in all the time if you don't want to tie up the extra cable and the 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 started 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.

Note: several more configuration options can be set by editing the config.h source file in the project and recompiling. See the section on config.h earlier.

Configuration settings made through the config tool are saved to the non-volatile (flash) memory on board the KL25Z. The settings will stick even after power cycling, controller resets, PC resets, and USB disconnects. However, updating the controller software erases the flash memory and resets everything to factory settings.

Here are the options you can set through the Windows tool:

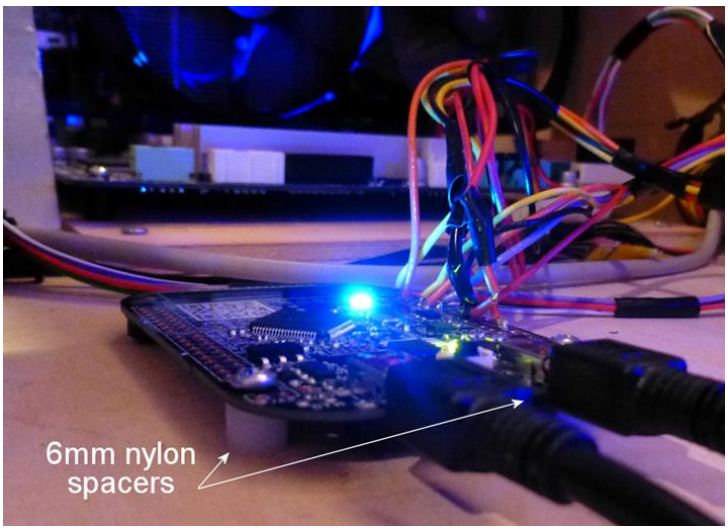
- Enable or disable the plunger sensor. The sensor is enabled by default. You can leave this enabled even if you don't attach the sensor – the software will simply see random pixel values if no sensor is attached, and will otherwise function normally. However, it might slightly increase the rate of accelerometer and key press reports sent to VP, because it takes a certain amount of time to read the sensor input on each cycle, even if no sensor is actually attached. (This won't affect the quality of the accelerometer data, since the firmware samples the accelerometer at a fixed rate regardless of the USB report rate.)

- 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. 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 default orientation is with the USB connectors towards 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 change the direction, you must edit config.h** (see above) and set the ORIENTATION_PORTS_AT_xxx option that matches the mounting direction.



Note that it's okay if your cabinet floor is slightly tilted relative the floor. This is the norm for pinball cabinets, so the firmware expects that there will be a tilt, and compensates by calibrating its idea of

"down" whenever the accelerometer is at rest for a few seconds. Simply mount the board flat on the floor of your cabinet, and the software will figure out the slight tilt of the cabinet floor and subtract it out from its reports.

It's important to fix the KL25Z firmly in place. The accelerometer has to move in lock step with the cabinet in order to experience the same accelerations as 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. Attach it with #4 x 1" screws (for wood or sheet metal) through the holes and 6mm-tall nylon spacers between the board and the cabinet floor. It should feel locked in place without any wiggle room.

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 selected the appropriate ORIENTATION_PORTS_AT_xxx option in config.h. That will ensure that the accelerometer readings reported to VP are oriented correctly for VP's X and Y joystick axes. Note that previous versions of this guide said to adjust the axis directions in Visual Pinball to match the orientation. You no longer need to do this. Starting with the August 2015 version, you should set the orientation with config.h, and use the default VP settings shown above in all cases. This new approach has the advantage that it should also work with other pinball player software, such as Future Pinball, that doesn't have the flexible option settings that VP has.

The "Enable Nudge Filter" checkbox is in VP 9.9.1 and VP 10 (and later). When enabled, the filter applies some special processing to the accelerometer input to make the effect in VP more realistic. The filter **isn't** the standard Physmod 5 version, but I created a custom version of Physmod 5 that has it. You can download it (plus a similarly tweaked build of VP 9.9) from the Pinscape project page on mbed.org. Note that my custom builds don't have the checkbox; the filter is simply always on.

Choose a plunger sensor

The Pinscape software can currently work with two different types of plunger position sensors: the TAOS linear CCD, and a linear slide potentiometer.

The software also works without any plunger sensor at all. I personally wouldn't dream of building a virtual cabinet without a plunger, since it's such a definitive element of pinball, but the software doesn't require one. If you don't attach any plunger sensor, the software will simply report constant 0 values for the plunger position. VP will still let you use a Launch Ball button to control the plunger as always.

Assuming that you want a real plunger, though, let's look at the sensor options and their pros and cons.

The CCD is my preferred sensor, for several reasons. First, it's highly precise; its resolution is as good as what you can see happening on-screen. Second, it's immune to calibration drift, because it senses position relative to pixels that are fixed in space. Third, there's no mechanical contact between the sensor and the plunger, so there's no wear and tear from use, and the sensor doesn't have any effect on how the plunger moves or feels. This also makes the mechanical installation simple.

But the CCD does have a few disadvantages. It expensive, and it requires a bunch of soldering work for the electrical connections. Some people have had dead-on-arrival experiences with this part.

The potentiometer is cheaper, and the electronic portion of the installation is simpler. The downside is that a pot has to be mechanically attached to the plunger rod. That makes the installation a little harder mechanically. It also can affect how the plunger moves and how it feels to the player, since there's extra friction from the pot as you move the plunger. The pot might also experience wear and tear from use, which might change the calibration or reduce the stability or linearity of the readings over time.

If you have ideas for other types of sensors, let me know! The software is designed so that code for new types of sensors can be easily plugged in alongside the current options. I can't guarantee that I'll be able to develop the code for new sensor types myself, but I can provide technical support if you're willing to do the work, and I'd be happy to integrate the results into the main project's source code.

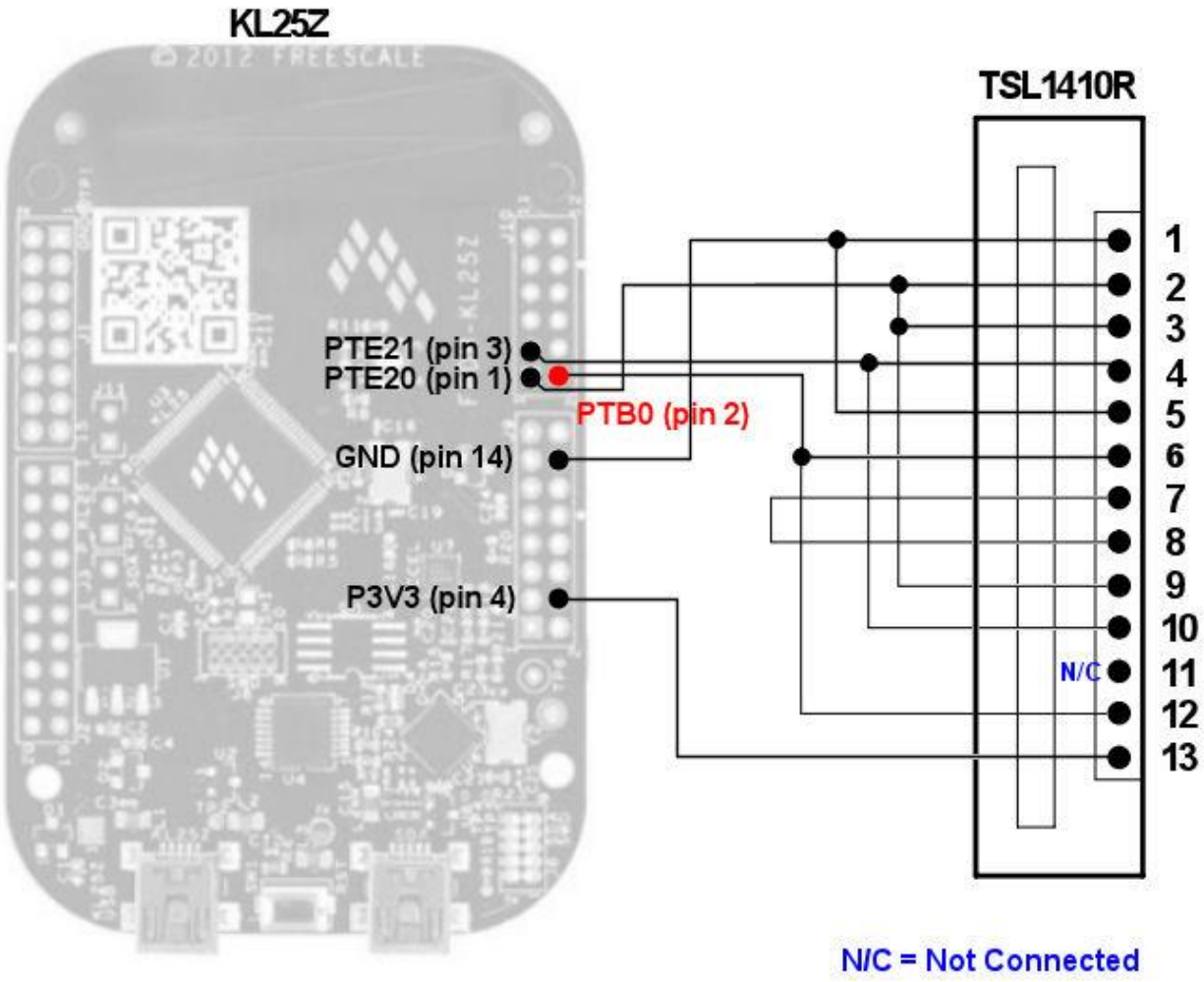
Wire the plunger sensor, part I: the CCD

The TSL1410R is a CCD photo-sensor array. The software uses this sensor to determine the position of the plunger by essentially taking pictures of the plunger rod with the CCD, and finding the tip of the plunger by analyzing the captured images. (The TSL1412R is also supported. This is very similar to the TSL1410R, but has a slightly larger pixel array. Update CCD_NPIXELS in config.h if using a 1412R.)

Warning! CCDs can be damaged by static electricity. Use precautions for static-sensitive devices when handling the CCD. Ground yourself before handling the device by touching a bare metal surface on a plugged-in PC or other electrically grounded appliance, and avoid carpeted floors and other sources of static build-up while working on the device.

The manufacturer's data sheet for the TSL1410R is available here:
http://www.taosinc.com/getfile.aspx?type=press&file=tsl1410r-e29.pdf

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



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.

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.

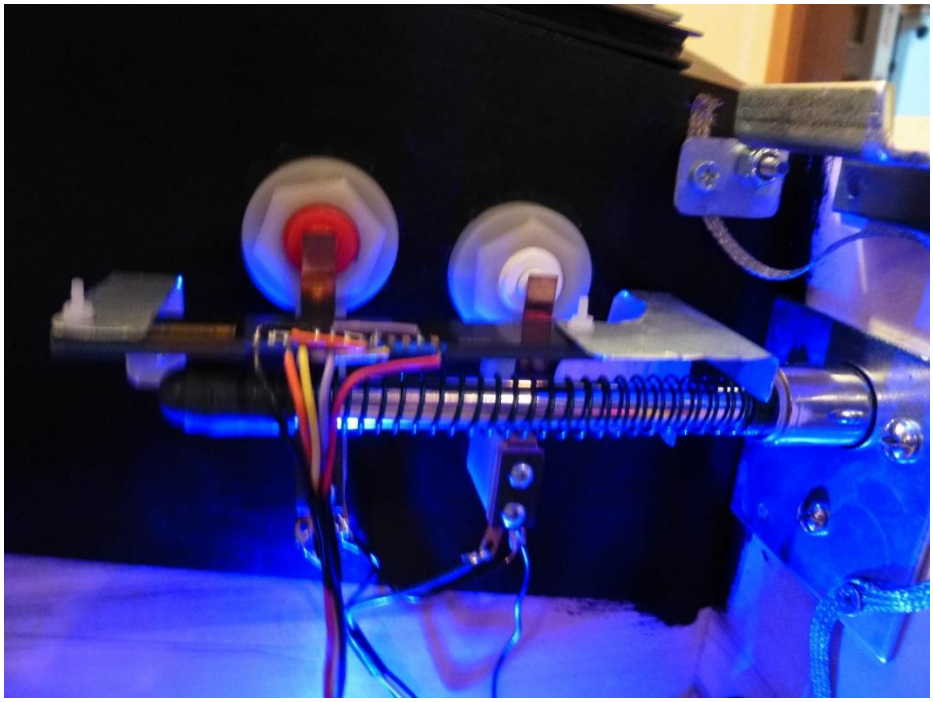
You might prefer to use pin headers and plugs (such as the ones in the parts list in the LedWiz emulation section) to connect to the KL25Z. That would be tidier than what I did. The main reason I soldered wires directly to the KL25Z is that the sensor requires connections to both the J9 and J10 jumpers, 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. If I built a new one, I'd find some way to use pin headers and plugs, though – I really prefer everything in my cabinet to be modular and easily pluggable.

Mount the CCD in your cabinet

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 the sensor is tolerant enough that it's really not 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 – about 1cm seems to work well
- The sensor's 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. 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 every reading to determine the light level range for the reading. 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 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.

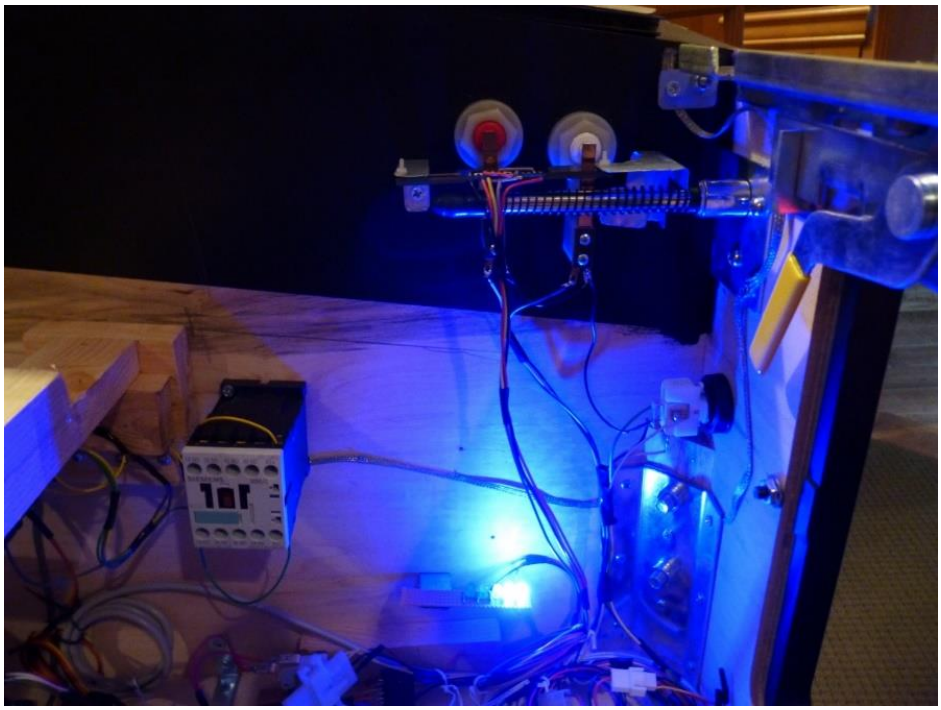
Note that the plunger can travel about half an inch forward from its rest position if you push it and compress the barrel spring on the front. It will do this naturally when you fire it with full force, so it's important to take this into account for the sensor positioning.

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 a light source for the CCD

The principle of operation of the plunger sensor is to detect the edge of the shadow cast by the plunger against the sensor. We want as sharp a shadow as possible. The shadow is sharper when the light source is further away, since it makes the light more of a point source.

The best lighting setup seems to be a small dedicated source positioned about 8" away. My setup, which has been very reliably, uses 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.



Note that I don't use any sort of shade or shroud around the sensor or the light source. The cabinet itself seems to make a good enough darkroom. There are stray photons from various other LEDs inside the cabinet, but these are apparently dim enough that they don't create any interference.

In my testing, the CCD has proved to be tolerant of a wide range of lighting conditions, particularly on the low end. Too much light will overexpose the sensor and prevent the software from detecting the shadow edge, but it seems almost impossible to have too little light. In my tests, the sensor could (almost unbelievably) work with just the meagre ambient light inside my cabinet from indicator LEDs on the motherboard and other devices. It's twitchy running this way, so you certainly shouldn't rely on that as your actual light source, but it's a good indication of how sensitive this device is.

Initial CCD 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 is tolerant of low light levels, but can easily get oversaturated by too much light. Make sure that the top side of the sensor isn't getting hit by 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.

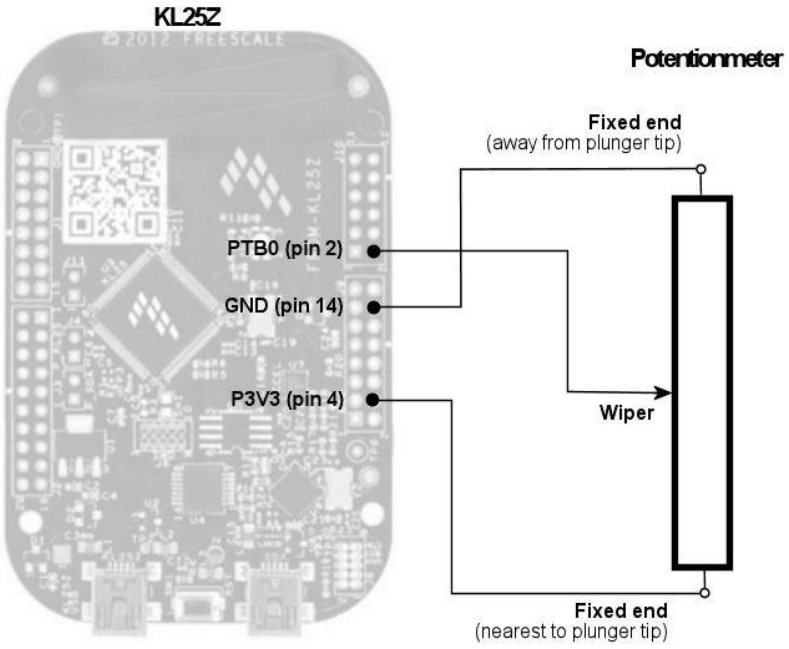
If the raw pixel status looks good, the next thing to check is that everything is working at the USB joystick level. Bring up the Windows control panel called "Set up USB game controllers" and open the Pinscape Controller. The X/Y position on the joystick display should respond to cabinet nudges – that's 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.

Wire the plunger sensor, part II: the potentiometer alternative

If you want to use a potentiometer for the plunger sensor, you'll need a "slide" potentiometer. This type has a lever that moves across the length of the device. The lever's travel distance should be 80mm or more, since that's how far the plunger can move. You'll need a **linear taper** pot. ("Linear taper" means that the resistance varies linearly with the position of the lever. The other kind is audio taper, where the resistance varies logarithmically. The software only works with the linear type.) The rated total resistance should be about 10K ohms.

Your potentiometer should have three terminals. Two are the fixed ends of the resistor, and the third is the "wiper", which moves across the resistor to produce the varying resistance. The resistance between the two fixed terminals will always be constant. The resistance between the wiper terminal and either of the fixed terminals varies as you move the lever back and forth. (If you're not sure which terminal is which, check the device's data sheet or test it with an ohmmeter.)

The KL25Z wiring is simple. You'll need three wires between the pot and the KL25Z. First, connect KL25Z ground (jumper J9 pin 14) to one of the fixed-resistance terminals on the potentiometer. Second, wire KL25Z V3.3 (jumper J9 pin 4) to the other fixed terminal on the pot. Finally, wire KL25Z port PTB0 (jumper J10 pin 2) to the wiper terminal on the pot.



Mount the potentiometer sensor

The basic idea here is straightforward. We just need to connect the lever on the potentiometer to the end of the plunger tip, so that the lever moves with the shooter rod. Then we mount the potentiometer itself to the cabinet to hold it in place.

The details of the mounting are up to you. You'll have to come up with something suited to the particular potentiometer you're using. To give you an idea, here's a picture courtesy of lemming77 from vpforums.org showing his setup. The white plastic housing is his own custom 3D-printed design; you can download the Sketchup file from the Pinscape page on mbed.org.

Note that the direction of the potentiometer matters. Mount it so that the resistance **increases** as you pull back the plunger. This *usually* means that you should mount it with the fixed-end terminal located at the tip end of the plunger. But these devices vary, so check the specs for your device, or test it with a voltmeter. (Or just try mounting it in a random direction and see what the software does. If the on-screen plunger motion is backwards, flip the pot around the other way.)

Testing and troubleshooting the potentiometer

Before you proceed, make sure you customized your version of the software for the potentiometer sensor option. See the section earlier in this guide about **config.h**. Edit config.h on mbed.org to comment out the line that says `#define ENABLE_CCD_SENSOR` and un-comment the line that says `#define ENABLE_POT_SENSOR`, then click Compile to build and download your customized firmware. Install that on your KL25Z using the drag-and-drop procedure described earlier.

Once you have everything installed and wired and you have the potentiometer-ized version of the firmware installed on your KL25Z, go to the Windows control panel and open "Set up USB game controllers". Double-click 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 (the right side of the scale).

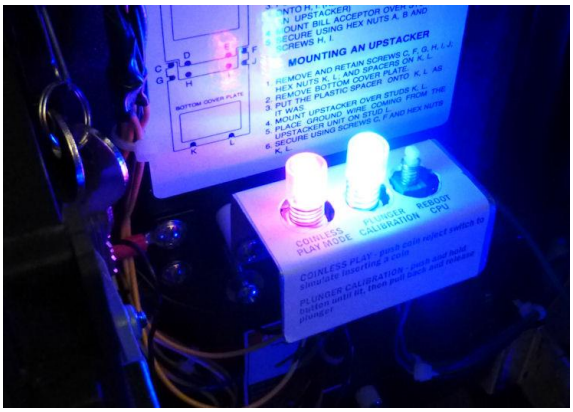
If the Z axis moves the opposite direction, you have the pot mounted backwards. Just flip it around.

All sensor types: Install the plunger calibration button

Okay, we're done with the multiple tracks for different sensor types. This step applies to any type of plunger sensor. But it's optional. If you don't install a calibration button, you can use the Windows configuration tool to activate calibration mode. That's not quite as spiffy, but it works just fine and saves you the trouble of installing even more hardware.

On my cabinet, I mounted the calibration button on the inside of the coin door, in a little cluster of my own special-purpose service buttons. My extra buttons are: 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); the plunger calibration button we're discussing in this section; and a PC Reset button wired to my PC motherboard, to do a hard reset on the PC if Windows should ever freeze. (Amazingly, I haven't had to use this a single time in over a year of working on this project. Hooray for Windows 7!)

I found some pushbuttons with built-in indicator LEDs that resemble the ones Williams used for the 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 exactly what I wanted.



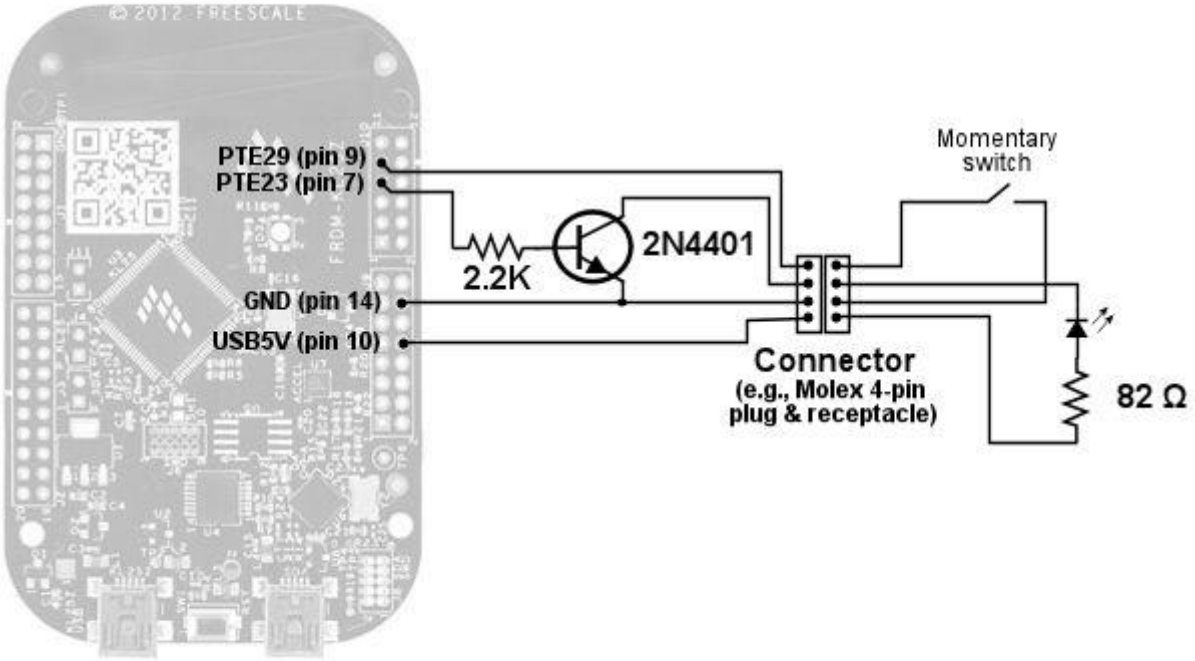
Left: the inside of my cabinet's coin door. My added service buttons are in the bright cluster in the upper right. This space was designed for a dollar bill acceptor, which I don't use, so I repurposed it for this.
Right: a close-up of my custom buttons. The middle button, glowing blue, is the plunger 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 family members and guests aren't always 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 the parts list and construct the circuit shown below. The transistor is needed to boost the tiny 4mA output current from the KL25Z pin to the 20mA level needed to power the LED. I built this part of the circuit on a 1cm x 1cm perf board and 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. There's a reason behind this. The transistor circuit will work with any indicator light (up to about 600mA), but the 82Ω resistor is sized for the particular LED in this particular button. If you should ever change to a different switch or indicator, the resistor value might change with it, so I grouped it physically with that part of the circuit. That makes the whole thing a little more future-proof. Electrically, of course, it makes no difference which side of the connector the resistor is on.

Calibrate the plunger

The plunger sensor (whichever type you're using) will need to be calibrated when first installed. This lets the software measure the range of motion and find the park position (where the plunger sits when at rest), so that it can report an accurate zero point and an accurate retraction distance 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.

Set up the Pseudo "Launch Ball" button

Zeb of zebsboards.com came up with a clever feature for his plunger kit that lets the plunger double as a Launch Ball button for plunger-less tables like *Medieval Madness* or *Terminator 2: Judgment Day*. The Pinscape software borrows Zeb's idea to offer the same feature. This section explains how to set it up.

Here's how the feature works. For plunger-less tables, when you pull back and release the plunger, the Pinscape controller software simulates pressing the Launch Ball button. It does the same thing if you just push the plunger in slightly, essentially treating the plunger knob like a pushbutton.

How does the controller know that the table is plunger-less? That's the clever part. Zeb came up with the idea of a fake LedWiz feedback device to signal that we're playing such a table. This fake device is now standard in the DOF Configtool – it's the one called "ZB Launch Ball". The DOF table list enables this in the appropriate tables. The Pinscape controller is an LedWiz emulator, so if you assign ZB Launch Ball to a Pinscape LedWiz ports, the controller will be able to tell when this output is turned on. And of course the Pinscape controller also can send button input to Visual Pinball.

To set this up, you have to configure the Pinscape software, Visual Pinball, and DOF:

1. Edit the Pinscape controller configuration file, config.h, as described earlier in this guide. Find the section titled "Pseudo 'Launch Ball' button". There you'll find definitions for the LedWiz port number for the ZB Launch Ball output, and the joystick button number for the Launch Ball input. The defaults are to use LedWiz port 32, and joystick button 24. If necessary, change these to suit your setup. Note that output port 32 is a good setting for almost everyone, because this port is usually not even attached to a physical output pin – there aren't enough pins on the KL25Z to fill out all 32 ports. And if you're not wiring any physical buttons to the KL25Z, any button input will do for the virtual one, so 24 is as good as any. But if you are wiring physical buttons through the KL25Z, you should change button 24 to the one where you physically wired your Launch Ball button, if you have one. If you don't have a physical Launch Ball button, just use any button number you're not using for a real button.

2. Open the Visual Pinball editor, and bring up the Preferences | Keys dialog. Find the Plunger item. Open the drop-down list of button numbers underneath the item. Select button 24 (or whichever button you chose above if you changed the default setting in config.h). Note that if you also use keyboard input, you can also leave the Enter key (or whichever other key you use) enabled. VP will happily handle both joystick and keyboard buttons for the same control.

3. Open the DOF Configtool in your browser. (If you don't use DOF, you really should. If you haven't set it up yet but plan to do so later, make a note to come back to this step when you do.)

    a. If you haven't already added your Pinscape Controller to your cabinet configuration, go to the My Account page. Set "Number of FRDM-KL25Z Devices" to 1. Click Save Settings.

    b. Go to the Port Assignments page. Select the "FRDM-KL25Z 1" item in the Device drop-down. Open the drop-down for Port 32 (or for the output port number you assigned above, if you changed the assignment in config.h). Set this to ZB Launch Ball. Save Settings again, and then click Generate Config to create and download your new DOF .ini files. Copy the .ini files to your DOF folder.

To test, open a plunger-less table like *MM* and try launching the ball with the plunger. Note that for this to work, the table has to be configured to use DOF. I won't cover that procedure here since that's a whole separate software package. Google it or ask on vpforums.org if you need pointers.

Modify your Virtual Pinball tables to work with mechanical plungers

Some of the Visual 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.

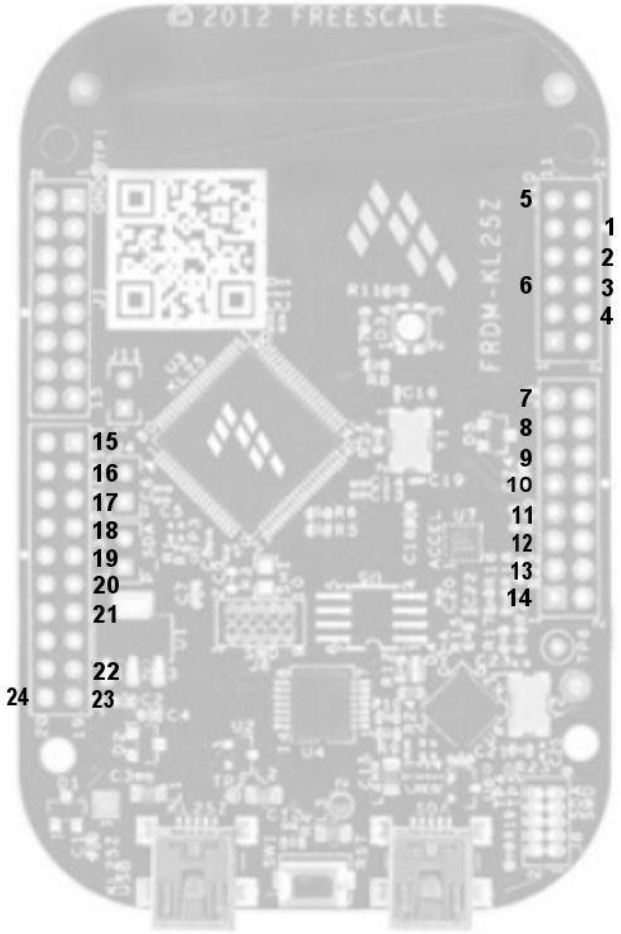
Once you've physically wired your buttons, you'll need to configure them in Visual Pinball. To do this, open VP without any table loaded, and use the Preferences > Keys menu to open the keyboard dialog. Each function (Left Flipper, Right Flipper, Start, etc) has a drop-down list underneath. For each 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 configuration file (**config.h**) 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



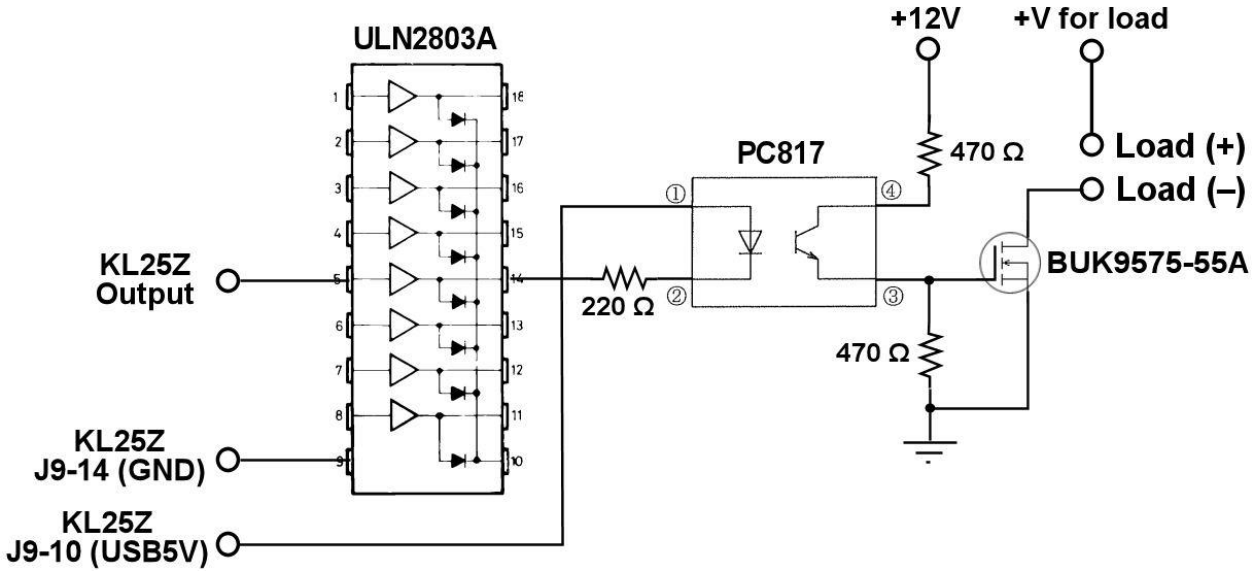
Build the LedWiz output drivers

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 specified in the shopping list can handle up to 20 Amps and voltages up to 55V DC. This will easily handle enough current for a shaker motor, replay knocker, contactor, or even a flipper coil. It's fast enough to pass through PWM signals,

so it can control the brightness on flasher LEDs or the speed of a shaker motor. (There's no need with this design for a "dual H bridge", which you might see mentioned in the forums for motor speed control.) 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 your power supply against short circuits or malfunctions in the feedback device.)

One big caveat: don't hook up anything involving AC power to this circuit. This type of MOSFET circuit is strictly for controlling DC devices. If you need to attach an AC device, put a relay on the output, and switch the AC with the relay.

Here's the schematic for my output driver circuit. 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. I use about 10 of these circuits in my own cabinet to drive various devices, some quite high-power, and it's been working nicely.



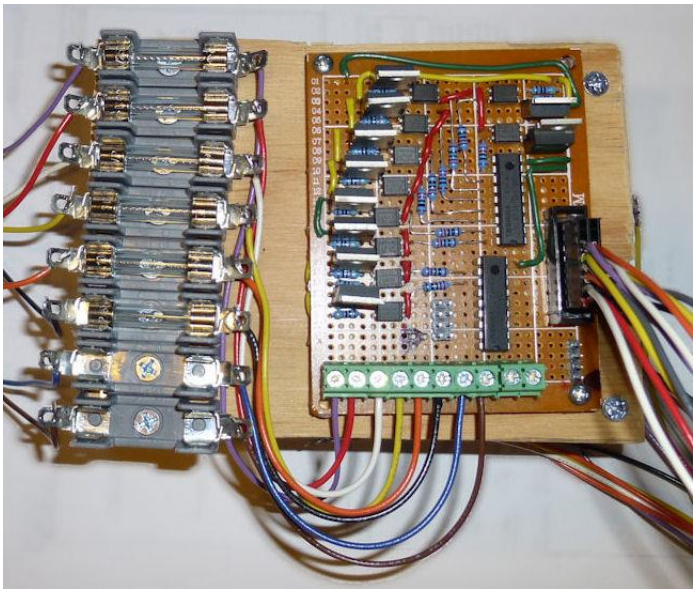
The input and output pins shown on the ULN2803A chip are for illustration only. This chip has 8 identical blocks, arranged with the inputs and outputs on the corresponding pins on opposite sides of the chip. You can therefore use one ULN2803A to build 8 copies of the circuit shown above. Just wire each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

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

Here's how the circuit works. The ULN2803A amplifies the signal from the KL25Z output pin (On = +3.3V, Off = 0V) 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.

For a relatively small load, such as a 20mA LED (or a few of them), you can wire the load directly to the ULN2803A output, omitting the optocoupler and MOSFET. The ULN2083A can handle 500mA per output, so it's safe to drive small loads directly. For anything bigger, use the whole circuit as shown.

Here's a picture of my driver board. You'll notice that there are two ULN2803A chips, for 16 outputs, but only 10 optos and MOSFETs. The reason is that I run 6 channels directly from the Darlingtons, for my flipper and Magna Save button lights. Each color channel on the flipper buttons has four 20mA LEDs attached (two per side), for a total of 80mA. That's well under the 500mA limit for the Darlingtons. The MOSFET outputs are driving a fan, a gear motor (the beefiest thing in my system, surprisingly – it draws about 4A), a couple of police-type beacons, strobes, undercab LEDs, and a couple of contactors.



The circuit above is only one possible way to drive high-power loads with the KL25Z. Virtual pinball builders often use relays instead of solid-state drivers, because they're perceived to be cheaper and simpler than MOSFET circuits and can handle plenty of current. I prefer MOSFETs because they don't click, don't suffer from mechanical wear and tear, and can handle PWM signals. And they actually can be cheaper than relays. The only real downsides are that the circuitry is a little more complex to assemble, and the MOSFET circuit above can't be used with AC loads.

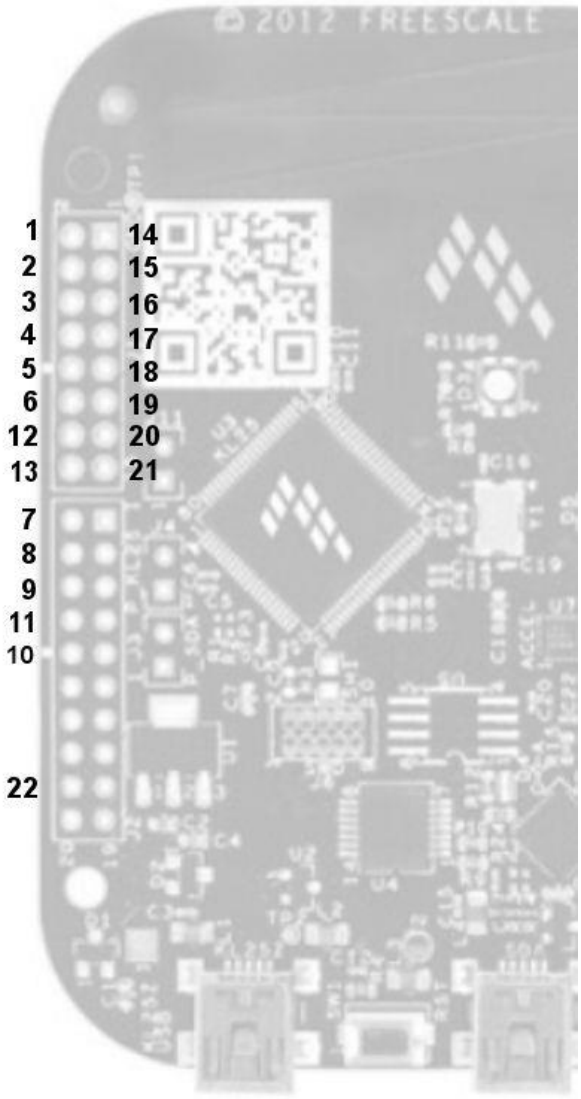
Note that if you do want to get the KL25Z to control a relay, you'll still need at least the Darlingtons, because the KL25Z outputs can't supply the current needed for even the smallest relay coils. You can drive a relay from a Darlington output as long as the relay's coil activation current is well below 500mA. Remember to put a flyback diode across the relay coil.

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. In the list below, **Ports 1-10 are PWM-capable.** Other ports can only be 100% on or off.

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	-	-



You probably noticed the LedWiz port numbers aren't arranged in simple sequential order on the physical jumper pins. I'd like to have arranged it that way, but it wasn't possible to do that *and* keep the PWM ports grouped together in the LedWiz numbering, which seemed more important. 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 bit to stay within the KL25Z limits.

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. I had to budget them among the various features (plunger, LedWiz emulation, and key inputs). I figured that it's difficult to use this controller as a primary LedWiz replacement because of the limited PWM port set, so most people who use the LedWiz emulation at all will only use it for a few additional ports, in which case it should be okay that we have less than the full 32. But if you really do want the full 32 output ports, it's possible if you're willing to sacrifice some button inputs. If so, you can take pins that are assigned to buttons in the default configuration, and reassign them as LedWiz ports. You can find instructions for reassigning pins as LedWiz outputs in comments in the configuration file (**config.h**). Look for the **ledWizPortMap[ ]** array definition.

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. 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 let the host 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 might not matter. It also doesn't matter for devices like contactors and knockers, where you'd never use a flashing mode in the first place.

Future work

Here are some ideas for future enhancements to the project.

• More types of plunger sensors. Other cabinet builders have come up with other clever ideas for plunger sensors. I like the CCD sensor quite a bit, but it does have some trade-offs, particularly its price. Some interesting ideas I've seen are LVDTs (linear variable differential transformers, which use magnetic inductance to sense position), linear encoders (quadrature devices such as magnetic or optical encoders, which can sense positions very precisely and are commonly used in printers and robotics), and optical mouse encoders (although I'm not sure these are fast enough to keep up with the plunger release motion). I like solutions best that involve no mechanical contact with the plunger rod and that are inherently immune to calibration drift, and so far the only thing I've seen that fits that bill is the CCD, but some of these other ideas come close and might have other advantages over the CCD.

• External PWM controller chips. We could offload the LedWiz emulation feature to separate PWM chips, which would make it possible to break through the 10-channel limit of the current design and have as many PWM channels as we want. It would be relatively easy to create a 48- or 64-channel PWM controller this way, which might make the KL25Z capable of serving as the only controller device in a fully decked-out cabinet. This would require some new external hardware, but not much more than we already need for the power boost circuitry. In fact, a PWM controller would take the place of the Darlingtons, and would eliminate the need for separate current-limiting resistors on each opto (since the most common PWM controllers all provide constant current to their output channels), so we might end up with a lower part count per channel than we have now.