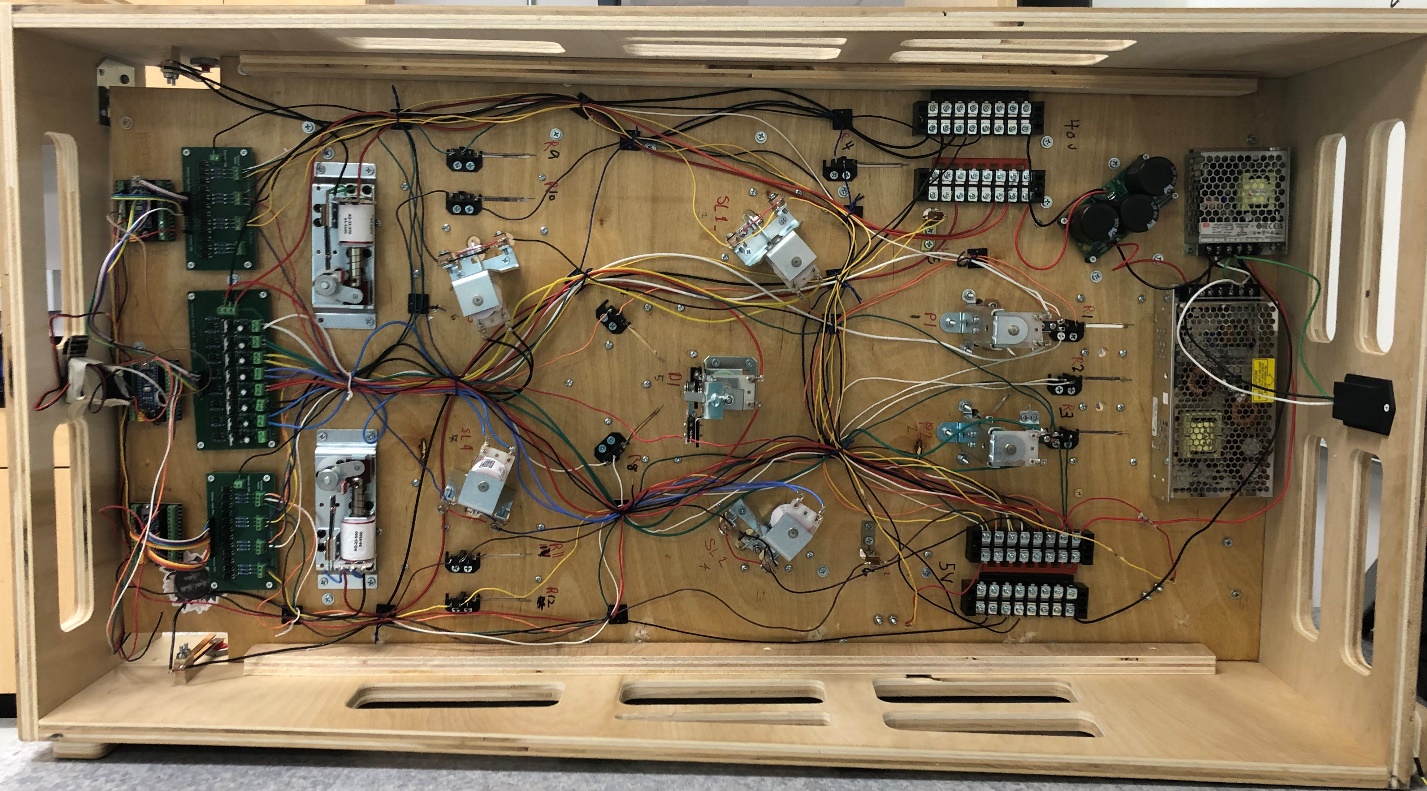
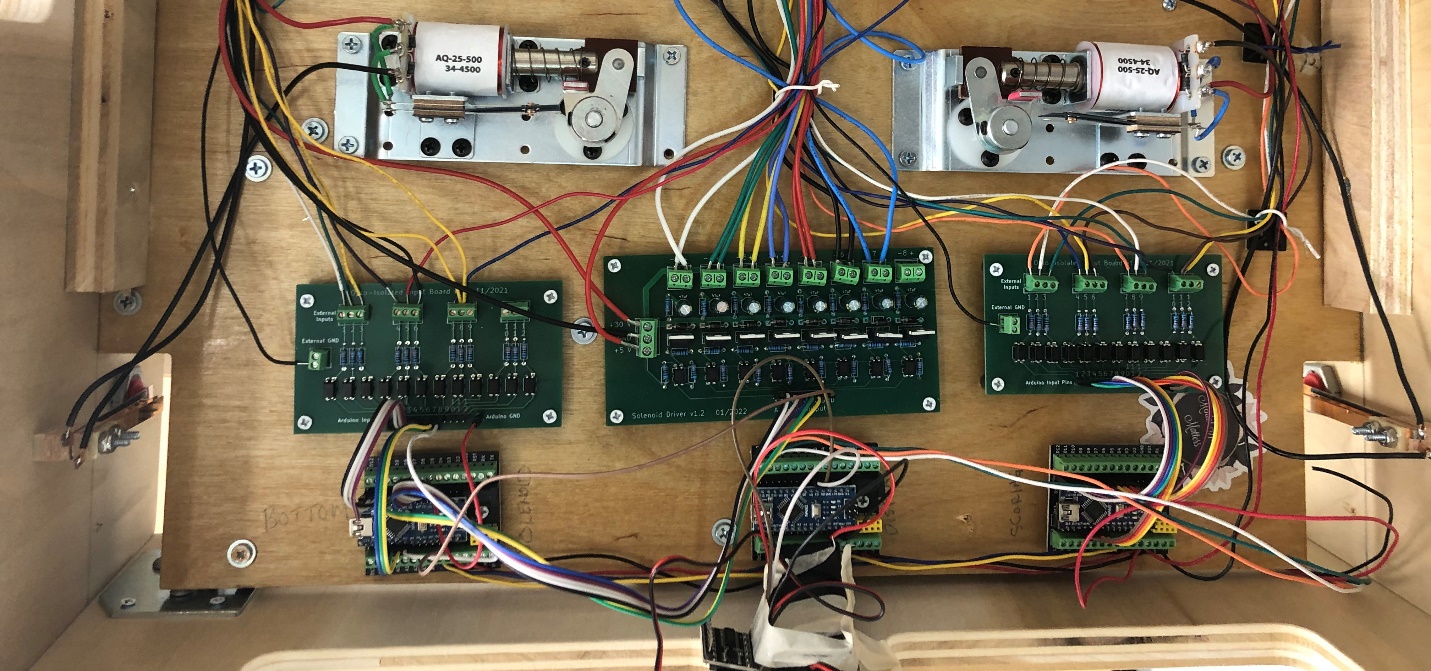
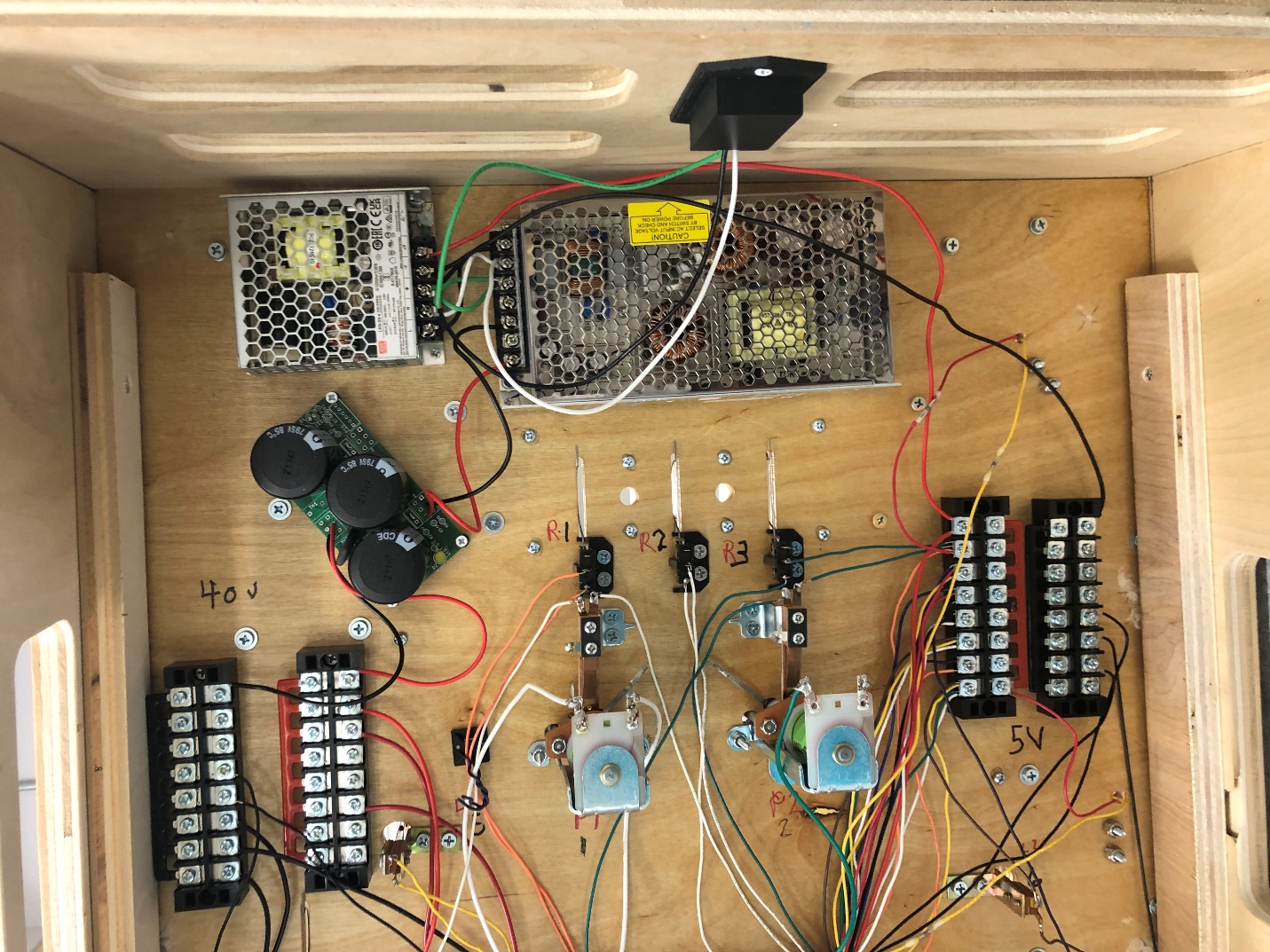
General Layout Hints for Electronics

Where you end up placing electrical components on the underside of the playfield will vary, depending on the playfield elements you included in your design. Below is a photo of a typical board created by my class. The power supplies and terminal blocks are typically mounted at the top of the board. The control electronics (Arduino Nanos and printed circuit boards) are mounted at the bottom. Despite your best efforts, the wiring will probably turn out to be a rat's nest. Don't stress out about it – just keep the wires away from the moving parts.



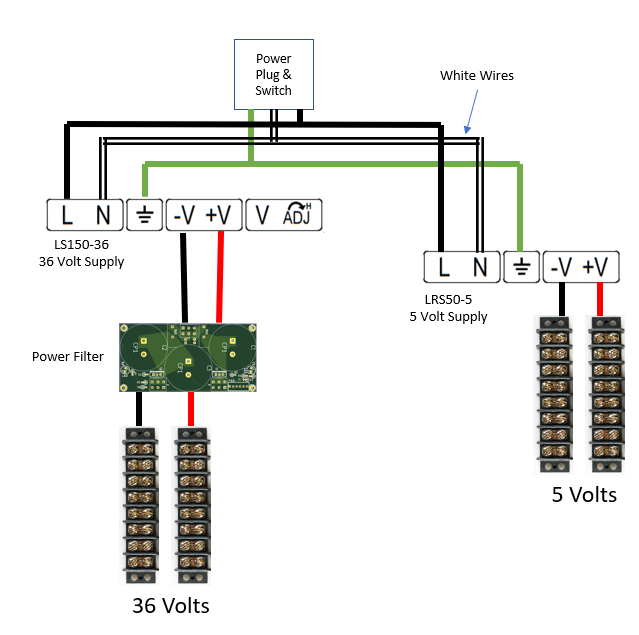




Wiring Diagram for the Power Supplies

**Parts List**

|  |  |
| --- | --- |
| TDK-Lambda LS150-36 Power Supply | Mean Well LRS-50-5 Power Supply |
| LS150-36 TDK-Lambda Americas Inc | Power Supplies - External/Internal  (Off-Board) | DigiKey |  |
| Power Filter Board | Power Plug & Switch |
| Power Filter Board |  |
| Terminal Block (4) |  |
|  |  |

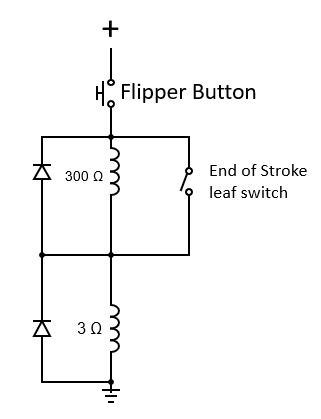


Notes on Wiring the Flippers

The flippers are not controlled by the Arduinos, they are connected to the power bus and the flipper buttons according to the schematic below. The solenoid coil contains two windings connected in series. The high-power winding has resistance of ~3Ω and draws a lot of current, providing the power for the “bat” to strike the ball hard and send it up the playfield at speed. The low-power coil has a resistance of ~300Ω and is initially shorted out. When the flipper button is operated, the high-power coil is energized from the 36-volt supply and the plunger enters the coil and forcefully rotates the mechanism. Towards the end of the plunger's travel, the End-of-Stroke leaf switch is opened, removing the short from the low-power coil. This reduces the current through the two coils but holds the mechanism in position until the flipper button is released. The above sequence prevents the high-power coil from overheating and possibly burning out if the flipper button is held closed.

The End of Stroke leaf switches on my flippers were not initially positioned correctly – they stayed in contact too long and shorted out the power supply – a leaf needed to be bent forward so the leaves are in contact for only a short time when the flipper is energized.

The contact surfaces of all leaf switches need to be lightly sanded with 800-grit sandpaper before use to ensure good conduction of electricity.



The Arduino Hardware and Software

The pinball machine contains three Arduino Nanos, one each for controlling the solenoids, handling non-solenoid elements, and displaying the score. These computers are connected in a simple network, formed by wiring all the A4 pins together and all the A5 pins together, allowing them to share information. This design was chosen for three reasons:

1. A lot of GPIO pins are needed, and using three Nanos is cheaper than using one Arduino Mega. You will probably end up frying at least one Arduino at some point and replacing a Nano is cheaper than replacing a fancier Arduino.
2. Using multiple computers reduces the latency of responding to events on the playfield.
3. The code for controlling the solenoids is quite tricky and should not be messed with. If you leave a solenoid energized for too long, it will melt. (I know this from experience.) Isolating this code on a separate computer prevents introducing software bugs that lead to hardware failures.

The solenoid Nano is connected to one opto-isolator board and one solenoid board. The opto-isolator board connects to the switches on the playfield mechanisms (e.g., slingshots, pop bumpers, and drop-down targets). One side of each switch is connected to +5V and the other side is connected to the opto-isolator board. The solenoid board controls the current flow to the solenoid coils. A description of the GPIO pins available for use with the software can be found in the file "Pinball\_Arduino\_Pin\_Assignments.xlsx". The most important thing to understand is that the switch associated with a solenoid #1 must be connected to opto-isolator input #1, the switch associated with a solenoid #2 must be connected to opto-isolator input #2, etc. You will need to edit the "main\_solenoid.h" header file to provide some information about the mechanicals attached to the computer. There are comments in the header file that provide instructions on how to do this. The solenoid computer informs the scoring computer when switches are closed. [I do not recommend modifying the software for this computer, unless you really know what you are doing. The one use case requiring software modification is if you want to add rules controlling when a drop-down target is reset. For example, you might want it to stay down for a set period.]

The scoring Nano is connected to one opto-isolator board, which connects to switches that are not associated with solenoids (e.g., roll-over switches, stand-up targets, and leaf switches). This computer is also responsible for calculating the score. A description of the GPIO pins available for use with the software can be found in the file "Pinball\_Arduino\_Pin\_Assignments.xlsx". You will need to edit the "main\_scoring.h" header file to provide some information about the mechanicals attached to the computer and how many points are associated with each. There are comments in the header file that provide instructions on how to do this. This computer sends information to the display computer. You can modify the software for this computer without worrying about melting anything or causing a fire.

The display Nano is responsible for driving all output. The software currently supports us of an 8-digit seven-segment display for showing the score, a DF Mini MP3 player for sounds, and one strand of NeoPixels for lighting effects. The GPIO pin assignments are described in the Excel file. There are plenty of pins still available if you want to add additional hardware. You will need to modify the software if you want to do anything spiffy with lights and sound.

Two sets of software are provided – one set is the vanilla code that has not been tailored to a specific machine design, and the other set is the actual code for a machine that includes a drop-down target that resets on a timer, plays sounds for specific events, and has various lighting modes that are play-dependent. Use the second set to give you ideas about how you might program your own play rules.