Author

Topic: Prototype Whitewood Project (Read 23737 times)

JoeShabadu2000

Wizard

Posts: 118

Prototype Whitewood Project

« **on:** August 26, 2012, 09:44:57 PM »

So, I have decided to build a custom pinball machine using the P-Roc system. The first question I have gotten from anyone I have told this to is, "What is the theme?" Well, I have decided to put off the theme for the moment. First, I want to gain an understanding of how a pinball machine goes together from an engineering (mechanical, electrical, and software) standpoint. Then, once I know what I can do with it, I think it will be easier to design a game that I am happy with. So, with that in mind, I am going to start implementing standard pinball systems (flippers, slingshots, pop bumpers, lighting, etc.) on a prototype whitewood, one at a time. My plan is to document the process as I go.

A bit about my background, since it seems like many of the people here are a lot more experienced than I am. I don't work in technology, but I have been working recently with microcontrollers (the PICAxe especially) and doing some basic work with robotics. I own 3 pinball machines, and have torn them down and rebuilt them, but I wouldn't consider myself an expert on pinball machines by any means. I don't know much at all about woodworking or construction, but I think with Youtube and persistence (and a willingness to redo something if I have to) I can make a go of it. With all that said, I think I have enough experience at this point to take this project on, so hopefully with the help of the community here we can work our way through all the issues that come up. Hopefully other "beginners" like myself can use this guide to help guide them through the process of putting a custom machine together, maybe make things a little less intimidating for someone else starting from scratch.

I have ordered the P-Roc starter kit (the P-Roc, one 8x8 Matrix controller for LEDs, and one 16 channel power driver) and some Molex connectors, I'll talk more about that piece once I get those items in. What I have focused on this weekend is getting myself a playfield blank, and a way to set it up to work with it.

I decided to construct a playfield rotisserie using the instructions found at http://homepinballrepair.com/pinballrotisserie.html. This design uses some galvanized pipe fittings, sawhorses, angle iron, and 2x4s to hold up the playfield, and allows you to rotate it over to work on the bottom side. I was able to get all the parts for this at Home Depot. I had to buy some sawhorses myself since I didn't have any, and I don't know that the ones I bought were ideal for the job, but I think they will work. Part of my idea is to have one sawhorse higher than the other, that way I can get a playfield angle so I can do some rudimentary playtesting while it is still on the rotisserie. So one of the sawhorses has adjustable legs. Based on my calculations, one side needs to be 5-6" higher than the other to give the standard 6-7 degree playfield incline. However, If I was doing it again, I would probably just buy the brackets that let you build a sawhorse using 2x4s, that way I could cut them to length and get whatever height I want on each, and avoid the issue of having to use the clamps that I currently am using.

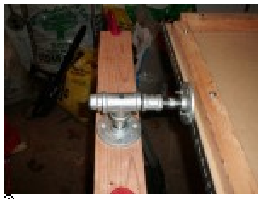
I tried to come up with my own method of locking the playfield so that I wouldn't turn when I didn't want it to, but it didn't really work well. I drilled some holes in the T fitting and the pipe inside, but I don't think I executed very well and there is still a lot of play when I put my locking pin in. The author of the guide I referenced above mentions using a threaded thumbscrew, that may be the way to go. I may end up just making some supports that I can put under the playfield that I will move aside when I want to rotate it.

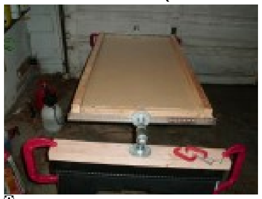
For the playfield itself, I am using a 4' x 2' x 1/2" sheet of MDF that I got from Home Depot for about $10. Real playfields are 17/32" thick, but that thickness is tough to find. That extra 1/32" helps to keep the top of the playfield from dimpling when using standard 1/2" wood screws to affix playfield components to the underside, but I think if I use a washer I should be cool. Most plywood labeled as 1/2" is actually 15/32", but the MDF is a true 1/2" thickness. MDF isn't something you would want to use for a production playfield since I'm pretty sure it is going to dimple once I start putting pinballs on there, but it seems like it will be hard enough for prototyping purposes. The sheet I got seems nice and straight, so I feel pretty good about it, and MDF seems pretty easy to work with. The playfield is attached to the rotisserie using #8 x 1 1/4" machine screws and bolts, they are easy to take on and off if I need to move the playfield to a new location.

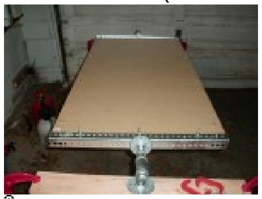
The dimensions for a standard Williams playfield are 20.5" x 46". To replicate those dimensions on my 24" x 48" piece of MDF, I have attached some wood strips along the top and sides. Now, the inside dimension (roughly) matches the Williams playfield. Plus, once I get some flippers installed, this will make it so that the ball doesn't go flying off the playfield. I will need to add some sort of bottom rails that will funnel the ball to a drain hole and a trough assembly, but that is down the road a little bit. The 2" x 2" lumber I am using for the "rails" is something that I had sitting around in my garage, and they aren't very straight. I don't think this is going to be a big issue, theoretically at some point I will be able to fabricate some metal rails that will mean the ball never touches them anyway. The wood rails are attached to the playfield using #10 x 3" machine screws and bolts. I clamped the rails down where I wanted them, then I drilled through the rails and the MDF in the same plunge, that way I could be sure all the holes and such lined up.

My next step, once I get the P-Roc in, is to get it (and the driver boards) communicating with my computer so that I can program the P-Roc and receive data from it. Then, I'll try to document the process of making the necessary connectors to attach the P-Roc and driver boards. After that, flippers, slingshots, and skies the limit!

CIMG0507.JPG (173.29 kB, 1024x768 - viewed 740 times.)


CIMG0508.JPG (171.03 kB, 1024x768 - viewed 660 times.)


CIMG0509.JPG (173.19 kB, 1024x768 - viewed 520 times.)


CIMG0510.JPG (150.35 kB, 1024x768 - viewed 515 times.)

Logged

Steve S
FPGA_testers
Posts: 434
Steve Shoyer

Re: Prototype Whitewood Project
« **Reply #1 on:** August 27, 2012, 12:18:08 AM »

Sounds like a cool project, seriously starting from a blank canvass.

I built a rotisserie using Mike's design (with Eli's one-ended modification) - his design is archived from RGP. It's basically the same design as the one you're using, but with black pipe rather than sawhorses. The first one I built used a bunch of 2x6 lumber that I had sitting around, which was OK but not very flexible. With my second one, I used some long pipes that were sitting around (I use them for woodworking clamps), so I didn't need to buy too many new components to use Mike's design.

To keep the playfield from flipping over, Mike's design uses a carriage bolt with a faucet handle that fits into a drilled/tapped hole in the pipe that acts as a sleeve for the axle. It works very well.

With your plan to have the rotisserie on an angle, I'd be a little nervous about spinning the table while still angled. If the axles are parallel to the ground, rather than on a straight line through the table, it might not spin too well. I guess you could put the sawhorses on a sheet of plywood as a base and raise up one end so the table (and everything else) tilted at 6-7 degrees.

Do you think you will eventually wind up using a standard sized cabinet (to fit your standard sized playfield), or will you build something custom?

--Steve

Logged

JoeShabadu2000
Wizard
Posts: 118

Re: Prototype Whitewood Project
« **Reply #2 on:** August 27, 2012, 11:40:24 AM »

One of the sawhorses I bought has individually adjustable legs. I plan to use this one as the "top" of the playfield, so I should be able to make the back legs longer than the front, which will give me the angle I need. For the "bottom" of the playfield saw horse, I think I'm going to just use some shims under the front legs to give them a bit of an upward angle, with the back legs sitting directly on the ground. My hope is that all of this will allow me to have everything lined up properly, but it's going to be an experiment for sure and I may need to change things up as I go along.

If this thing ever gets placed in a cabinet, I do intend to use a standard size cab. That is one of the advantages to using a standard playfield size. My thought right now is to maybe buy a flat pack cab (such as http://virtuapin.net/index.php?main_page=product_info&cPath=2&products_id=78) and assemble it, but I could potentially use an empty cab from another game if I got one at the right price.

Logged

Snux
Wizard

Posts: 781
Mark Sunnucks

Re: Prototype Whitewood Project
« **Reply #3 on:** August 28, 2012, 12:54:48 PM »

Looking forward to following along with your progress! How are you going to cut out the inserts, holes etc? Do you have access to a nice routing machine or will you do them all by hand?

Logged

F14 Tomcat - Second Sortie

JoeShabadu2000
Wizard

Posts: 118

Re: Prototype Whitewood Project
« **Reply #4 on:** August 28, 2012, 01:35:01 PM »

For this first prototype, I'm going to be doing everything by hand. My credo is to measure twice, make a precise mark, and to drill pilot holes using a small bit to get everything as accurate as possible. That being said, I'm sure it isn't going to turn out perfectly aligned, but hopefully it will be close enough for my purposes. My woodworking skills are minimal, so I see a good amount of frustration and wood filler in my future, but my hope is that I will come out the other side with more skills than I had to begin with.

Once I get in to the "real" design phase, I think I'm going to have to use CNC, both for accuracy and repeatability. There is a "hackerspace" in my city that has a CNC router available (along with lots of other tools), eventually I'm going to have to move a part of my operation there.

Logged

JoeShabadu2000
Wizard

Posts: 118

Re: Prototype Whitewood Project
« **Reply #5 on:** August 30, 2012, 09:34:56 AM »

Alright, so the P-Roc arrived, on the same day as my Antek power supply. I also got my order in of connectors and my crimper, so it is time to start getting the hardware set up.

First, I prepared my ATX power supply. I am using a cheap "420 Watt" supply, which is about 10 years old at this point, I think it is fair to say it is about 300W in actuality. I have been using this as my 5V and 12V supply for my other electronics projects, so I am just going to reuse it here. ATX power supplies by default don't turn on when you apply power, you have to hit a momentary switch (usually attached to a motherboard) which makes them activate. This is a pain, so what you can do is attach a jumper wire between two of the pins on the large connector (see picture below for details). Now, whenever I flip the switch on the back of the power supply, it starts right up.

The P-Roc has a power input that attaches directly to the 4 pin connector from the power supply (J26), so no modifications were needed there, just plug in and go.

To get power to the PD-16 and Matrix 8x8 boards, you will need to get a +5V line and a ground line attached to a 2 pin Molex connector, size 0.156". On a standard ATX power supply, the +5V comes off the red wire, and ground is the black wire. To access these voltages, I cut off the last 4 pin connector from the same chain I am using to power the P-Roc. I trimmed down the yellow (+12V) and one of the black wires flush with the previous connector so the wires won't get in the way or send stray voltages anywhere. For the remaining red and black wires, I left them about 5" long and stripped off about 1/8" of insulation from each wire. Then, from my spools, I cut two 5" lengths of hookup wire, one red and one black. I stripped about 1/8" of insulation from each end. An automatic wire stripper, such as the one available here for $19, makes this job a snap. The wires are now ready for crimping.

First, you crimp on a contact to the exposed wire, then insert the contact into the appropriate connector. I am using the BCT-1 crimper (aka 1026-CT from Sargent Tools), available from Great Plains Electronics for $28. This step also requires the following items (also available from Great Plains).

08-52-0072	Crimp Contact, 0.156"	$0.08
09-50-8021	Connector, 0.156" Plug, 2 Pin	$0.12

You will need 4 contacts and 2 connectors to make the necessary power connections for the PD-16 and Matrix 8x8. I twisted together the exposed end of my +5V wire from the power supply and the red jumper wire, and crimped both of those in to the same contact. I then did the same for the ground wire from the power supply and the black jumper wire. Then I crimped one contact each on the remaining exposed ends of each jumper wire.

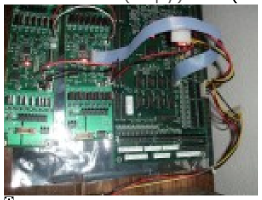
Insert the contacts in to the connectors, remembering that Pin 1 is the +5V input and Pin 2 is ground. The contacts only fit in to the connector one way, so if you have to force it then you are doing it wrong.

Now plug all the power connectors in to their appropriate receptacles. Turn on your ATX power supply. If all is well, you should see a nice LED glow coming from each of the boards. See the PDF manuals for each board to get a description of what they all mean.

At this point you can turn the power back off and hook up some of the data cables between these three boards. The P-Roc starter kit comes with the 16 pin cable you will need to connect the P-Roc to your "Master" board (in my case the PD-16), just connect those two together.   The connector on the P-Roc is J34. You will also need to connect the "Serial Out" from your master board to your other board using 0.1" connectors. For this part, I had some 0.1" female to female breadboard jumper wires sitting around, so I just used those. If you don't have these handy, you can order the appropriate contacts and connectors from Great Plains.

That's all for now, next step is software installation so we can communicate between the PC and our P-Roc.

CIMG0513 (Copy).JPG (273.29 kB, 1024x768 - viewed 731 times.)


CIMG0514 (Copy).JPG (308.89 kB, 1024x768 - viewed 568 times.)

CIMG0515 (Copy).JPG (152.03 kB, 1024x768 - viewed 553 times.)

CIMG0516 (Copy).JPG (171.71 kB, 1024x768 - viewed 598 times.)

Logged

Gerry Stellenberg
Administrator

Posts: 2399

Re: Prototype Whitewood Project
« **Reply #6 on:** August 31, 2012, 01:50:49 PM »

Thanks so much for providing the detailed instructions. Perhaps we you're complete, or at least farther along with your project, we can move the information to a wiki page.

- Gerry

Logged

JoeShabadu2000
Wizard

Posts: 118

Re: Prototype Whitewood Project
« **Reply #7 on:** August 31, 2012, 04:43:33 PM »

No problem, I hope people find it useful! I figure that I am going to spend enough of everyone else's time trying to help me figure this thing out, I might as well spend some time helping other people get through some of the issues I had. I'll just keep posting my updates here, if there is anything you need me to do to get this ready for the Wiki or whatever just let me know.

Also, if you see anything I have written that is incorrect (or if I have an incomplete understanding) please let me know. I'm going to start writing soon about the software, which is my weakest part (out of many weak parts), so there are bound to be errors coming up. Thanks.

Logged

JoeShabadu2000
Wizard

Posts: 118

Re: Prototype Whitewood Project
« **Reply #8 on:** September 01, 2012, 08:21:14 AM »

Time now to get our PC set up to start developing with the P-ROC. I am using Windows XP (32 bit) on the machine that I have hooked up to the P-ROC, and I also have a Windows 7 (64 bit) system that I will also be doing some software development on. My goal is to be able to move python files between these two systems and have them behave in the same way without needing modification, and I think the instructions below should allow you to do that.

One quick note, some of the directory structures I am going to be describing may not be the same as the "Recommended" structures from the wiki or other posts, so please be aware that if you are going to follow my steps below, you will also need to be using the config.yaml file I will post below (or be aware of the directory structures at least). This is not meant to necessarily be a "best practices" guide, but I'm just going to post what works for me, and hopefully it will have some value for others as well.

Anyway, to install the software you want to follow the instructions on the Getting Started post, or at the pyprocgame setup guide on the wiki. That being said, I will go through the steps with a focus on Windows, and there are a few steps that I had to take that aren't included in those instructions before I was able to get everything working properly. My "unofficial" steps are denoted with an asterisk.

Step 1: Install Python 2.6. The file you want is on this page. Download the "Windows x86 MSI Installer (2.6.6) (sig)". This is the 32 bit installer version 2.6.6. Use this for Windows 7 64 bit as well, don't get the 64 bit version, the 32 bit version will install fine and will ensure compatibility.

Step 2 (a): Go to the P-Roc software page here and download libpinproc_1.1b_install.exe (or a newer version if available). Double click to install. On XP, everything should go smooth. On Windows 7 64 bit, after libpinproc installs, you will get an error stating that the installer is not 64 bit compatible. This is OK. At this point, you should be able to go to your "Add and Remove Programs" feature in Windows and see that libpinproc is installed.

*Step 2 (b) (may not be required on all systems): The libpinproc installer creates a directory in c:\Program Files\P-Roc (or c:\Program Files (x86)\P-Roc for Windows 7). Inside this directory is libpinproc\bin, which contains 3 .dll files (ftd2xx.dll, libgcc_s_dw2-1.dll, and libstdc++-6.dll). Copy all 3 of these files into the c:\Windows\System32 directory. If you are running Windows 7, also copy them in to the c:\Windows\SysWOW64 directory. This step is not included in the other sets of directions, but taking these steps fixed some .dll issues I had later in the process.

Step 2 (c): Go to the libpinproc\ext\python folder inside the P-Roc folder that was created in part a, and run the .exe file. This will install the python pinproc extension on your system.

*Step 2 (d): Add the directories for Python (and the eventual directory for procgame) to your Path so that you can just type "python" or "procgame" and have them execute from the command line from whatever directory you are in. On Windows 7, click the Start button and type "edit the system environment variables" and click the program that comes up. Click the Environment Variables button at the bottom of the window. In the lower of the two windows that pops up, scroll down to select Path, then choose Edit. Hit the "end" key, then copy the following in:

Code: [Select]

```
;c:\python26;c:\python26\scripts
```

Then click OK and close all the windows. XP is very similar, just go to the desktop, then right click on My Computer, then Properties. Go to the Advanced tab, click the Environment Variables button, and follow the same instructions as above. I don't believe this will take effect until after you have rebooted, so I would recommend doing that before you go on to Step 3. Be advised that Windows is very sensitive to the formatting when entering your path, don't include superfluous spaces or anything like that.

Step 3: Download pyprocgame here . Just click the link that says "zip" near the top of the page to download the entire directory. You will need to unzip this somewhere on your hard drive, for installation purposes it doesn't really matter where, but there are files in there that you will want to review later so don't forget where you put it.

This is a good time to talk about the directory structure and the home directory. In XP, your home directory is c:\Documents and Settings\[UserName]. In Windows 7, it is going to be under c:\Users\[UserName]. So, if your username is Shabadu, then your home directory is going to be c:\Documents and Settings\Shabadu\. In all the documentation, this is referred to as ~. At the command prompt in Windows, just type "cd %userprofile%" and it will take you right there.

I have created a folder in my home directory called P-ROC, and this is where all my python files go. So, in my case, I am unzipping the pyprocgame zip into the folder ~\P-ROC\pyprocgame\ (which is actually c:\Documents and Settings\Shabadu\P-ROC\pyprocgame\). Other files will go on the P-ROC directory later.

Step 4: Install pyprocgame. Go to a command prompt, change directories to the folder where you just unzipped pyprocgame, and type:

Code: [Select]
```
python setup.py install
```

If you set up your path correctly in Step 2(d), this should execute the setup program. This will also install the setuptools application, if you are connected to the internet.

Step 5: Install the remaining dependencies for pyprocgame. This includes PYYaml, PIL, and PYGame. Each of them has a Windows installer, be sure to use the 32 bit versions regardless of what version of Windows you are running. Always be sure you are getting the versions compatible with Python 2.6.

Step 6: Download proc-shared, just grab the zip file like you did in Step 3. I unzipped this to ~\P-ROC\shared, so once you are complete there should be 3 folders inside that ~\P-ROC\shared directory.

Step 7 (a): Create your .pyprocgame directory inside of your home folder. You have to do this from the command line, I believe. Just use the following code:

Code: [Select]
```
cd %userprofile%
md .pyprocgame
```

Step 7 (b): Create your config.yaml file inside of the .pyprocgame directory. The contents of this file took me a long time to figure out, but it really isn't that complicated. If you have put your files in the directories that I mention above, then the following config.yaml should work for you:

Code: [Select]

```
font_path:
    - .
    - ~/P-ROC/shared/dmd

# Uncomment the following line to use a virtual P-ROC instead of running on real
pinproc_class: procgame.fakepinproc.FakePinPROC

config_path:
    - ~/P-ROC/shared/config

keyboard_switch_map:
    # Enter, Up, Down, Exit
    7: SD8
    8: SD7
    9: SD6
```

You will notice that the line referencing FakePinProc is not commented out in this file, which means that our system will be configured to use the virtual P-ROC rather than the physical P-ROC. For testing purposes, we will use the P-ROC emulator, and once we have everything working good there, we will get our real hardware hooked up and change the config.yaml file so that line is commented out.

*Step 8: Testing. First, go to a command prompt, and type "procgame config". If you have everything set up correctly at this point, that command should show you the location of your config.yaml file. You may get some error about logging not being enabled, apparently this isn't a big deal. If you get .dll errors or something of that nature, then you have an issue. Post on the Software forum and hopefully someone can help you figure it out.

Just to be sure everything is cool, we are going to run Judge Dredd in our virtual P-ROC. Download the zip file containing the JD files here. Unzip them in to a directory called ~\P-ROC\JD. If you want, you can also download the media pack from here, but it is not required and is 200MB, so don't feel obligated. Extract these media files in to the JD folder you just created.

Now go to the command prompt, change to the JD folder you just created, and type "python jd.py". You should see some messages in the console about fonts being cached, and after a little bit you should see a virtual DMD window pop up that displays the attract mode for JD. You can press "s" to start a game, or "7" to go in to service mode (see the config.yaml description for the mapping).

So that is it for the software install, next edition is a little more software and some more hardware as we get everything connected to the P-ROC itself.

« Last Edit: September 01, 2012, 08:25:45 AM by JoeShabadu2000 »

Logged

JoeShabadu2000

Wizard

Posts: 118

Re: Prototype Whitewood Project

« **Reply #9 on:** September 01, 2012, 10:46:14 AM »

Before we hook up our P-ROC to the PC, we need to go back in to our config.yaml file and put a # in front of the line that references FakePinProc. If you don't, things aren't going to work right, and there will be much confusion and gnashing of teeth. I have attached an image to this post that may serve as a helpful reminder.

With that out of the way, we can get down to business. The object today is to make sure that the PC communicates with the P-ROC, and that the P-ROC communicates with the PD16 and Matrix 8x8. By the end of this post, we will have a test lamp hooked up to the Matrix 8x8 that flashes on and off.

Lets start with the hardware. If you set everything up as described in my Aug 30 post, you are already most of the way there. We do need to change some DIP switches. Remember for this example that we are using a PD16 as the Master board, which is connected to the P-ROC using a 16 pin cable, and the Matrix 8x8 is connected to the PD16 as the only board in the serial chain. At this point, all the DIP switches on the P-ROC should be off, and all the DIP switches on the PD16 should be off as well. On the Matrix 8x8, turn DIP switch 1 on, and DIP switch 8 on. DIP switch 8 terminates the serial chain, since our Matrix 8x8 is the last board in this series. Turning DIP switch 1 on (and leaving 2-4 off) gives this board an "address" of 1. On the PD16, we left DIP switch 1-4 off, which gives it an address of 0. This is discussed in the manuals, and this address will be used in our machine yaml file to tell pinproc where our lamps, coils, etc are attached. If you have more than 2 driver boards attached, you will need to start using higher addresses, but we will stick with 0 and 1 for now.

We also need to connect power to the "high power" sections of the Matrix 8x8. In my prototype, we are going to be using 5V as our source voltage for the LEDs. I will be using #44 and #555 LEDs with a rated voltage of 6.3V, but based on my tests so far I think that 5V gives me sufficient brightness, and we have 5V easily available from our power supply. We will need a 5V and ground for both bank A and bank B, which are connectors J2 and J6 on the Matrix 8x8. After chopping off the end of another chain of connectors from my ATX power supply, I cut, stripped, and crimped following a similar procedure to how I am supplying the logic power to the two driver boards, the only difference this time is that we are using a 3 pin Molex connector instead of the 2 pin connector used for the logic. Pin 1 is 5v, pin 3 is ground. The following is the Great Plains part number I used for the connectors:

09-50-8031	Connector, 0.156" Plug, 3 Pin	$0.18

When you turn on your power supply you should notice now that the LEDs next to the input power banks light up, these are LEDs D1 and D5. You should also see D14 light up on both the Matrix 8x8 and the PD16. This is normal at this point. Also, note the presence of the light on the P-ROC at D44 and the absence of a light at D43, this will come in to play later.

At this point, we still haven't hooked the USB cable up to the P-ROC. Go ahead and do that now while we still have the power on. Your PC should recognize that the USB device has been attached and any driver setup should take place if necessary. Once everything looks cool, go ahead and turn off the power to the P-ROC.

To test everything out, we are going to need to create two more files. One is going to be our basic starter code, which I am going to call "simple.py". The other is going to be our machine definition yaml file, which will be called "simple.yaml".

There are already some example machine definition yaml files in the ~\P-ROC\shared\config folder. The following code is based on the sample code that Gerry provided in the sample_pdb.yaml file, with a few very minor changes to account for the way we configured our hardware earlier in this post.

Generate a file in the ~\P-ROC\shared\config folder called "simple.yaml" and copy in the following code:

Code: [Select]

```
# P-ROC Game Description file a sample machine using PDBs (PinballControllers.com D
PRGame:
    machineType: pdb
    numBalls: 4
```

```
PRFlippers:
    - flipperLwR
    - flipperLwL
PRBumpers:
    - slingl
```

The only real changes are that the name of the first PRLamp has been changed to testLamp, and the address of the lamps driver board has been changed to 1 to reflect how we set it up earlier. The coils were already set to address 0 in the example, so we don't need to change those. Don't worry about the fact that we don't have any of this stuff hooked up yet, we will get there soon.

Create a new directory in the ~\P-ROC\ folder called "simple", and create a file in that directory called "simple.py". Paste in the following code and save your file:

Code: [Select]

```
import procgame.game
import pinproc
game = procgame.game.GameController(machine_type=pinproc.MachineTypePDB)
game.load_config('simple.yaml')
game.enable_flippers(enable=True)
game.lamps.testLamp.schedule(schedule=0x00ff00ff, cycle_seconds=0, now=False)
game.run_loop()
```

I'm not really going to try to explain how this code works right now, because my understanding at this point is very basic. I just know that this code will work for our testing purposes. As we get in to some more complicated code examples, and as my understanding improves, I will make some more posts on this topic. Also, if anyone reading this is interested in helping with this project and has a solid understanding of Python and pinproc, please send me a PM. To my mind, there is a pretty large jump between the complexity involved in the simple code above, and the code in the "starter.py" or even "creature.py" examples, and I think it would be really beneficial for people without much programming experience if there were some more basic examples available, adding features one at a time and building in complexity as we go along. I plan to follow this path myself, but it is becoming obvious to me that getting up to speed on Python is going to be one of my biggest hurdles on this project, and any help would be greatly appreciated.

Anyway, back to our regularly scheduled program. Turn on your power supply so that the P-ROC lights up, and make sure the USB cable is connected. Go to the command line, change to the ~\P-ROC\simple directory, and type "python simple.py".

If everything went correctly, a few things should have happened. In your command line window, you should see a message containing your P-ROC firmware number, and the program should continue to run until you hit Ctrl-C to exit back to the command line. On the P-ROC, D43 will come on and D44 will turn off. On the PD16 and Matrix 8x8, D14 should be turned off.

If some or none of those things happened for you, then there was an issue somewhere, use the LEDs to help you determine where your issue is. If D44 never turns off, then the computer isn't communicating with the P-ROC. If D14 turns off on the PD16 but not on the Matrix 8x8, then you may have an issue with your serial connection, or your DIP switches may not be set properly. Just make a post in the appropriate forum (P-ROC, Driver Board, etc.) with your issue and someone will try to help.

Assuming all is good at this point, let's get our test lamp flashing like I promised you earlier. Turn off the P-ROC for now. In the picture attached to this post, I am using a #44 lamp socket, and a 6.3V #44 LED that I got in an assortment from Pinball Life. Here are the part numbers:

077-5002-00   Miniature Bayonet Base 2-Lead Socket With Short Mounting Bracket $1.50
abl_bayonet_sample   Ablaze 1-LED #44/47 Bayonet Base Lamp Sample Pack $7.35

We want to hook the positive terminal up to pin 1 on Bank A (J7) and the negative terminal up to pin 1 on Bank B (J11). In my picture I am using red and black insulated alligator clips to make these connections, which is certainly not a "recommended" way to handle this connection, and it may not really be "safe", but I like to live life on the edge.

Turn the P-ROC back on, run your simple.py program, and if that LED is flashing then you are good to go at this point. If not, double check your FakePinProc setting in config.yaml, try reversing the polarity on your LED, and if those don't work then you might need to make a forum post.

That is all for now, next up is getting some flippers flipping.

IF YOU FORGET TO COMMENT OUT FAKEPINPROC BEFORE USING YOUR P-ROC
INSTRUCTOR
YOU'RE GONNA HAVE A BAD TIME
26014013.jpg (117.75 kB, 400x400 - viewed 465 times.)

CIMG0519 (Copy).JPG (273.07 kB, 1024x768 - viewed 603 times.)

Logged

Snux
Wizard

Posts: 781
Mark Sunnucks

Re: Prototype Whitewood Project
« **Reply #10 on:** September 01, 2012, 11:19:39 AM »

Hi Joe,

This is great documentation. As Gerry mentioned, at some point when the "getting up and running" piece is more or less complete it would be great to get this into the wiki. You are working through the same issues/errors/challenges that we all did; the only difference is that we didn't do much of a job of documenting them beyond some occasional cries for help in a forum thread.

I have a similar setup to you. I develop on an old Windows XP laptop. In my F14 pinball machine I have a PC running Windows 7 (actually Tiny7). When the PC in the pinball boots, it loads Python and then tries to run Python code which is on a USB memory stick. So if I update my Python code and want to make it a bit more permanent (ie running the pinball without a cable dangling out to my laptop) I just copy the new python code (and supporting DMD/sound files) onto the USB stick.

Keep posting; it's fun reading along!

Logged

F14 Tomcat - Second Sortie

Gerry Stellenberg
Administrator

Posts: 2399

Re: Prototype Whitewood Project
« **Reply #11 on:** September 01, 2012, 11:23:46 AM »

Great stuff! My only suggestion at this point is to stop cutting your ATX power cables. 🙂

You already have the necessary tools. You just need a few more connector components to build pluggable ATX power extenders or breakout cables. Unfortunately GPE doesn't have these. So I buy them from digikey.

To mate with the 4-pin 5/12v ATX output connector:
0.200" 4-position receptacle
0.200" male crimp pin

To recreate the 4-pin 5/12v ATX output connector (if desired):
0.200" 4-position plug
0.200" female crimp socket

Note - you can browse digikey's associated products (on each product page) if the housings or crimp terminals listed above aren't precisely what you need (ie. different wire sizes).

- Gerry
« Last Edit: September 01, 2012, 11:25:41 AM by gstellenberg »

Logged

JoeShabadu2000
Wizard

Posts: 118

Re: Prototype Whitewood Project
« **Reply #12 on:** September 01, 2012, 11:36:26 AM »

Gerry, I'm one step ahead of you on the ATX power cutting front, I actually felt bad doing that second one. I ordered some male and female pins and connectors from a seller on eBay. Unfortunately, they did not arrive before the weekend, and I didn't feel like holding out until Tuesday to continue my experiments. That being said, I don't think I am going to need to make any additional connections to my power supply at this point. Any additional "open" 5V power (such as powering short range opto transmitter IRs) is going to come from the 5V out on bank B of the Matrix 8x8, and the connections to the PD16 are going to be 70V/24V which will come from the Antek power supply, so I think I have done all the damage I will need to do.

I appreciate having the part numbers though, usually I have good luck with eBay sellers but I know DigiKey is a bit more reputable. 🙂

Logged

JoeShabadu2000
Wizard

Posts: 118

Re: Prototype Whitewood Project
« **Reply #13 on:** September 03, 2012, 10:45:01 AM »

Before we can get the flippers working, we need to get them some power. The Antek power supply that comes with the starter kit does not include an AC plug that would go in to your wall. The idea is that you can wire the connector for either 110V or 220V, so you have the option. Since I'm in the US, we are going to use 110V. Per the data sheet, for 110V, both red wires are connected to neutral, and both black wires are connected to hot. I purchased an AC plug from Lowes, I think it was about $5, I decided to use a reasonably heavy duty model since we are are going to be connecting two sets of fairly heavy gauge wires to each terminal. I stripped about 1/2" of insulation off each of the 4 wires coming from the power supply, twisted the reds together, twisted the blacks together, then inserted the red set in to the left side of the connector (silver screw) and the black set in to the right side of the connector (brass screw). Nothing gets attached to the ground (bottom green) connector in this example. Then just tighten the connector itself and you should have a good connection.

Since the wires on the power supply are so short, I just bought an AC extension cable so I had some room to plug it in to the power strip. Also, there is no switch on the Antek power supply, so what I have done is plugged my ATX supply and my Antek supply in to the same power strip, and I control the power to both using the switch on the power strip. When you have the Antek supply powered up, you should see the 3 LEDs on the 70V output light up. You will also note that the LEDs stay on for a few minutes after you disconnect power, this is because the large capacitors on the power supply are still charged. Be very careful when working with the Antek power supply, just because the power has been turned off doesn't mean that you can't get a good shock from the capacitors.

70V power comes out of one of the terminals on the Antek supply and in to connector J5 on the PD16 board. This power will be used to control coils on Bank A of the PD16, once we actually get some coils hooked up. We are using the same 0.156" contacts and 3 pin Molex connectors that were used to provide power to the lamps on the Matrix 8x8. The yellow wire in my photo goes from the + output of the power supply to pin 1 of J5, the black wire goes from the - output to pin 3 of J5. I have not daisy chained my connection from Bank A on to Bank B, like I did with the Matrix 8x8. At this point, I believe I am going to be using 70V on both banks of the PD16, but since the Antek supply provides 3 fused 70V outputs, I think it is better to connect Bank B directly to its own output from the power supply, that way we have separate fuses in the circuit all the way back to the power supply.

A quick note about my wire color scheme. Black is going to be used for any connections that go directly to ground. Red is going to be used for "uncontrolled" 5V lines (any 5V that is active at all times that the power is on). Yellow is going to be used for "uncontrolled" 70V lines. Switches and lamps will have their own color schemes, once I actually start wiring those up. I have some colored heat shrink tubing that I will use to differentiate different "chains" as necessary based on an idea from Chris and Nye, thanks for the inspiration! I got my colored 18 gauge wire, and my colored heat shrink tubing, from sellers on eBay, you should be able to find suitable examples if you search (or send me a PM). You don't have to adopt my color scheme on your build, but I would recommend having several different colors of wire, and using each color for a specific purpose, that way hopefully you can cut down on some troubleshooting down the road and keep yourself a bit more safe by knowing where the dangerous voltages are.

Astute observers will also note that I have created a mounting "plate" for my P-ROC and driver boards. This starts with a 12" x 12" sheet of 1/4" plywood. I just placed the circuit boards on the plywood, and marked the hole locations with a pencil. I drilled those holes out, then mounted the circuit boards using #4 x 3/4" machine screws and nuts (14 altogether). Between the circuit boards and the plywood, I used some 1/4" nylon spacers (available in the drawers in the screw sections at Lowes or Home Depot). The spacers keep the boards off the plywood and should provide a little bit of ventilation to the backside of the boards. You could just as easily use 1/2" or longer spacers if that is what you have available, just adjust the screw length accordingly. Now it is easy to move the boards as a unit and I don't have to worry about them sliding around.

The flippers are also going to require switches to activate them. On a real pinball machine, the flipper buttons are going to be mounted to the cabinet, but since we don't have a cab we are going to need to mount them to the playfield instead. The flipper buttons and switches are ones that I ordered from VirtuaPin.net. I am using leaf switches, they should be sufficient for testing purposes at minimum, and to me they actually have a pretty nice feel so I wouldn't be surprised if they make the cut for a "final" build. Here are the the part numbers I am using:

2 x Flipper Button (3A-7531-5) = $5.00
    Button Color Red
2 x Pal Nut - Nylon = $0.70
2 x Leaf Switch (VPCLeaf) = $7.00

The buttons themselves consist of two parts, there is a threaded shaft that is 5/8" in diameter, and about 7/8" long. Then, there is a smooth rounded part that is about 1 1/8" in diameter, and 1/4" long. The idea in a real build would be to use a piece of 3/4" plywood and countersink the larger diameter portion so that the lip is flush with the cabinet. We aren't being that fancy right now and I don't mind the button sticking out a bit, so we are going to ignore that extra 1/4" for the smooth part, and as such we will be using 1/2" plywood. I bought a 2' x 2' sheet at Lowe's for about $6.

Again, a note about plywood thicknesses. In the US, if you are buying regular plywood, the actual thickness is 1/32" less than the rated thickness. So, my 1/2" plywood is actually 15/32". If you are using MDF, the rated thickness and the actual thickness are the same. For these buttons, there really isn't a lot of extra depth once you get these things mounted, so I would recommend using plywood rather than MDF in this application. That 1/32" might really make the difference (but I haven't actually tested it). Since these buttons are designed to be mounted into 3/4" plywood (actual depth 23/32"), it makes sense to use regular plywood here.

I cut my 1/2" plywood into two 6" x 6" squares, one for either side. In the center of each square, I drilled a 5/8" hole, into which went the button, which is attached to the leaf switch assembly using the nylon nut. There should be just enough space to get that nylon nut threaded on to the button, but it will be tight. At this point, depress the button and make sure that the leaf switches are making good contact with each other, you may need to bend the leaf switches inward if not.

I decided to attach this board to the 2" rails I installed along each side, rather than trying to attach in to the side of the 1/2" MDF. The rails are about 1/4" in from the edge of the MDF, so I cut some strips of 1/4" thick plywood to 6" x 2" lengths to fill this gap. For each side, I clamped everything together using 3" c-clamps, then drilled three pilot holes through the 3 pieces of wood, and used three #8 x 1 1/2" wood screws to attach it all together. This seems to give me a reasonably stable surface for the buttons.

You will also notice that, per Steve's suggestion earlier in this post, I have added the "Faucet Handle" playfield locking mechanism, which works really well. I had to buy a new drill bit and a tap set, but I think it was worth the effort. It doesn't seem like it gives enough tension to make it totally rock solid (if I am going to drill along one of the edges of the playfield I might need some additional support), but it is way better than it was before. Thanks for the suggestion Steve!

CIMG0521 (Copy).JPG (270.89 kB, 1024x768 - viewed 571 times.)

CIMG0522 (Copy).JPG (231.31 kB, 1024x768 - viewed 595 times.)

CIMG0523 (Copy).JPG (156.77 kB, 1024x768 - viewed 498 times.)

CIMG0524 (Copy).JPG (156.13 kB, 1024x768 - viewed 494 times.)

« *Last Edit: September 03, 2012, 10:49:59 AM by JoeShabadu2000* »

Logged

Gerry Stellenberg

Administrator

Posts: 2399

Re: Prototype Whitewood Project

« **Reply #14 on:** September 03, 2012, 10:38:23 PM »

To hopefully avoid any issues related to diodes (hopefully I'm not too late), you have two choices with the solenoid wiring. The easiest option is to cut all diodes off of all coils. The PD-16 boards have their own diodes; so the coils don't need them. If you leave on the coil diodes, then you have to get the polarity correct. Power goes to the bar-end of the diode and the appropriate switched circuit (from PD-16 J7/J11) goes to the non-bar end. If you have diodes on your coils and hook up the polarity backwards, the diodes AND transistors on the PD-16 board will blow.

I've removed all diodes from all of the coils on my machine. It makes wiring easy.

- Gerry

Logged

SMF 2.0.4 | SMF © 2013, Simple Machines
XHTML RSS WAP2