A Programming Jig for our DougsWordClock.com DeskClock Boards

by **drj113** on May 30, 2013

Table of Contents

Author:drj113
I have a background in digital electronics, and am very interested in computers. I love things that blink, and am in awe of the physics associated with making blue LEDs.

Intro: A Programming Jig for our DougsWordClock.com DeskClock Boards

Hey - It has been *ages* since I submitted something to Instructables to give back to the community, so I thought I would share how I built our new programming rig for the boards that are used in the www.dougswordclock.com DeskClocks.

You know how it goes, you have created an amazing project and told lots of people about it and sure enough tons of people would like one for themselves. You get the circuit boards made, and spend ages applying solder paste and components, reflow soldering and installing the non-reflow components, then you start loading microcode into the controller so that the project does what it is designed to do..... Wow - What a lot of steps!

Loading the microcode????? Yep - as you may know, a computer without software is pretty useless. All of our clocks have a special program loaded into them so that they can tell the time correctly. When I build the boards for the clocks, there is no software loaded into the microcontroller chip (The chip is too small to put into a normal programmer) - This process puts the software into the chip so that the clock can function.

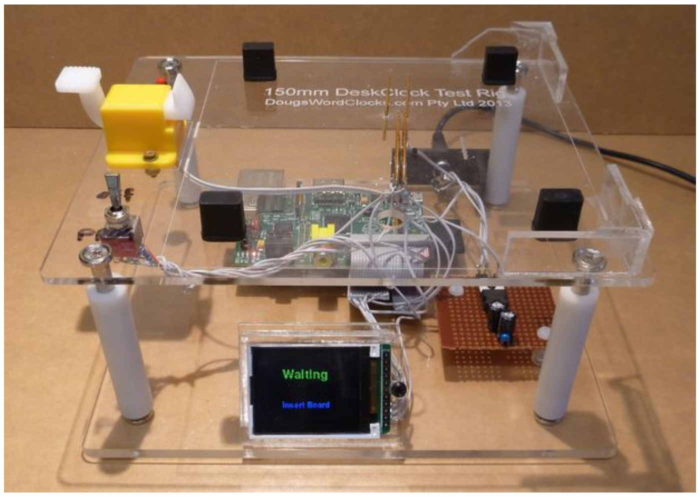
In the old days (a couple of weeks ago :-)) I used a laptop, a USBTiny programmer and the wonderful AVRDUDE software to program the boards - I would sit at my desk in the workshop, type the programming command into the computer, hold the programming cable against the clock and hit ENTER. The computer would then dutifully program the board for me and I would be done. The only catch with this is that I have to sit there all the time, so I decided that one of my employees could do it instead.... Unfortunately, he found that sometimes he moved the cable a tiny bit causing the programming job to fail and he would have to start again. To compound the issue, if there was a soldering error, the USB port of my laptop would shut down and the USBTiny would have to be unplugged and reseated to reset the USB port..... There had to be a better way!! how did the Big Boys do it?

It turns out that the Big Boys (tm) have robots that are very good at holding cables still and funky electronics that can do testing. Because DougsWordClock.com runs out of my garrage, I was not likely to get there any time soon, so what could I do to make our life easier? As my friend Mikal would say.... "Build a jig!". (Note1)

So here we have the Jig that Doug made! While it is specifically designed to program the DougsWordClock.com DeskClock boards, the concepts here can be extended to any other microprocessor based project you are building in bulk, so have a read about how I solved the problem and see what you can make yourself!

Lets start.

Note 1 - In early 2000, My best friend Mikal dropped into my workshop as I was building a set of shelves- I was routing joints, which was a boring repetitive task - Mikal said "Build a Jig!" I said "Too hard - I will be finished soon" - He said "Nahh, lets just do it" .... We did. Long story short, the simplicity of building a jig, coupled with the fact that I didn't think of it hit my Ego hard...I decided was useless....(Go figure).. Eventually, I snapped out of it and decided to write an article to prove to the world that I wasn't useless - So I designed a PIC based electronic dice project. It was even published by Silicon Chip magazine - (http://archive.siliconchip.com.au/cms/A_102324/printArticle.html) True story, and probably the start to me breaking back into using microcontrollers for projects and writing articles.. :-)



Step 1: The DeskClock Board

First, I started with the DeskClock board. When I designed it, I provided a 6 pin connector to allow a programming cable to be connected - Here is a photo of the board, showing the various connectors.

Of course - when we load components onto the board we don't populate these connectors - they are simply there for programming and testing.

The side in this photo of the back of the board, as opposed to the front of the board with all of the LEDs - it gets populated first. during manufacture.

I used this board to very carefully measure the location and spacing of the various connections I wanted to connect to.

Now - how did I connect to the board? Glad you asked. i used Pogo pins!

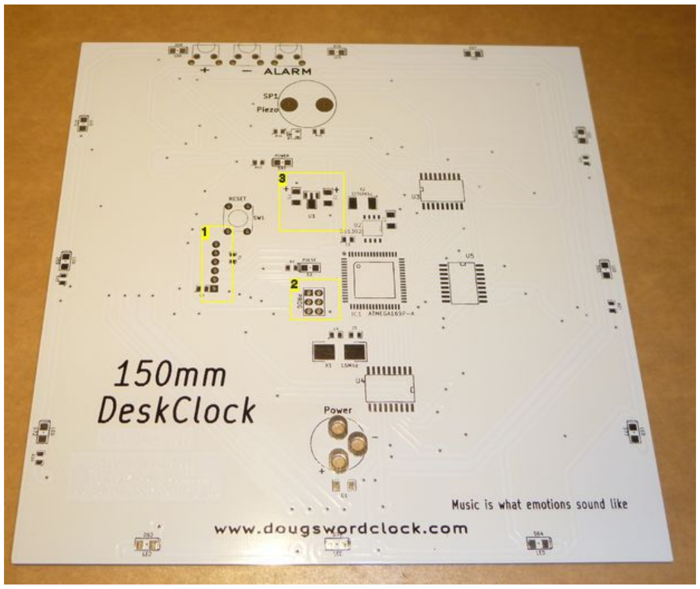


Image Notes
1. Serial Debug Header
2. Programming Header
3. Voltage regulator - More on this later

Step 2: Pogo Pins and other hardware

Pogo pins are what professionals use to make temporary connections to boards when they are testing them. They are available in lots of sizes and shapes, and have a precision spring loaded mechanism to ensure that the pin is pushed against the board with even pressure.

I brought my Pogo pins from a supplier on eBay - They were cheap enough that I think I have a life time supply now! The same supplier also provided me with the other tricky hardware that I needed to use to clamp the board down.

Here are a couple of photos of the pins themselves, the nifty board clamp, and the rubber board spacing mounts.

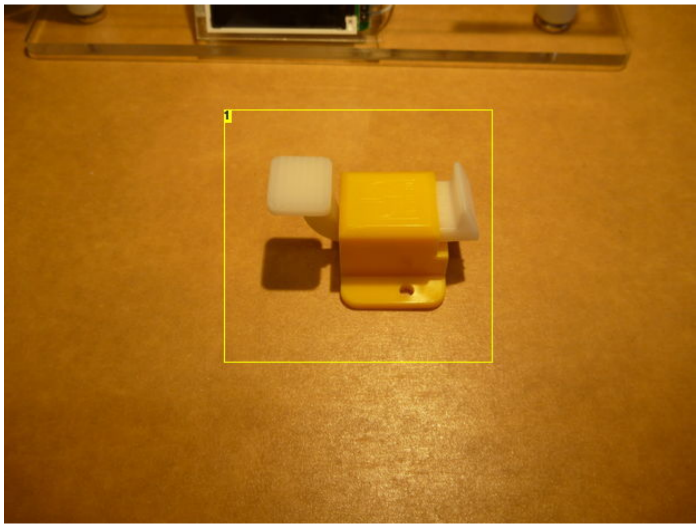


Image Notes
1. Board Clamp

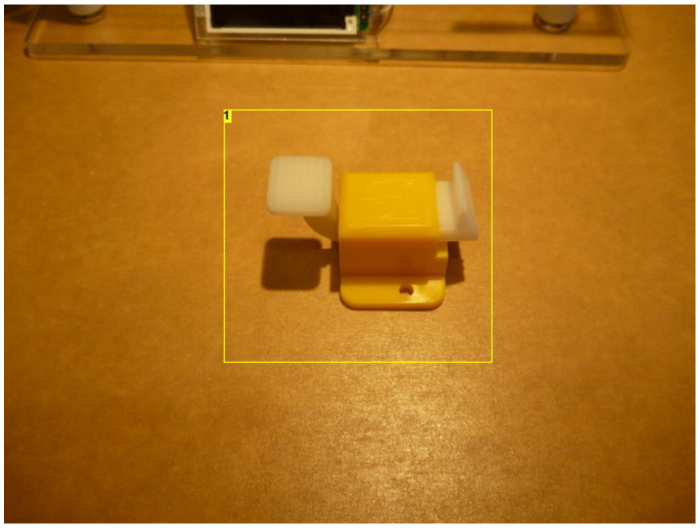


Image Notes
1. Enough Pogo Pins to last a lifetime!

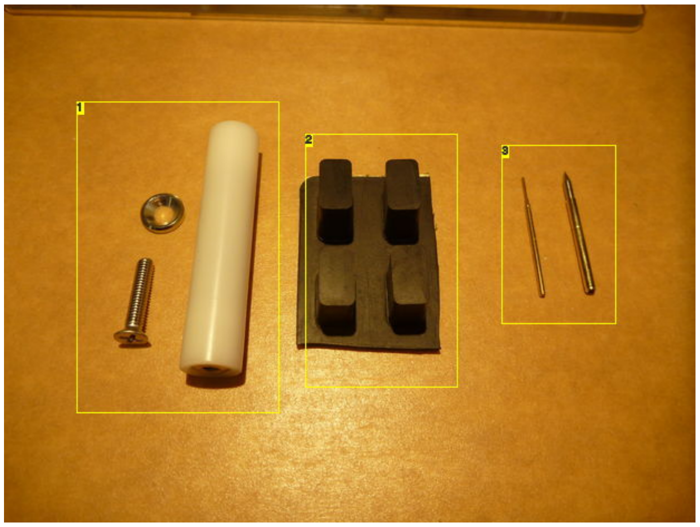


Image Notes
1. Vertical Support
2. 4 rubber board supports
3. Two different Pogo Pins

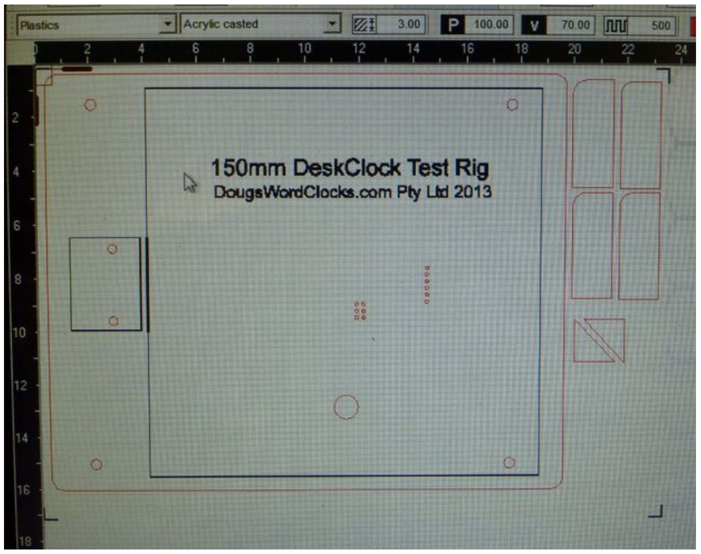
Step 3: Measuring and Making the Mount for the Board

So, I have the Pogo Pins and other mounting hardware. I carefully measured the hole sizes and spacing and created a layout for my Laser Cutter. i could also have simply drilled holes using a drill, but the cutter does a beautifully repeatable job.

I decided to space the holes so that the pins were not in the center of the pad holes - this ensured that the pins contacted the board firmly.

I also designed space for the Clamp and some tabs for the back of the board.

In the case of the DeskClock board, there is a 2.1mm axial power socket installed on the board that I had to provide a relief hole for, and finally, don't forget the rubber mounts to support that back of the board.



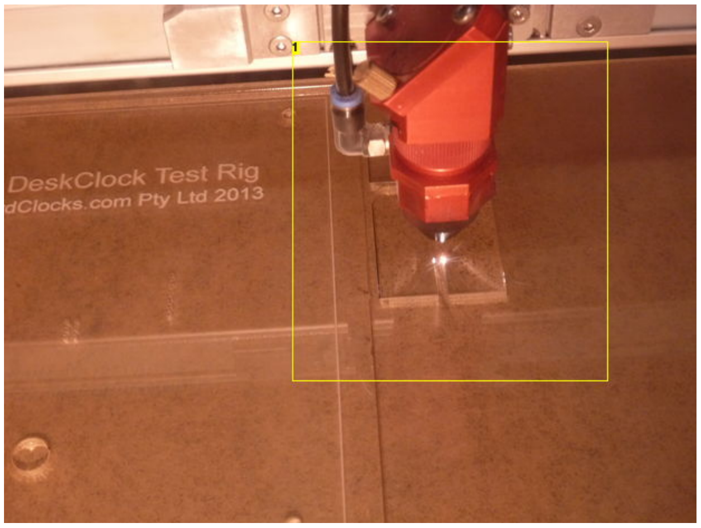


Image Notes
1. I love my laser cutter - It is so accurate :-)

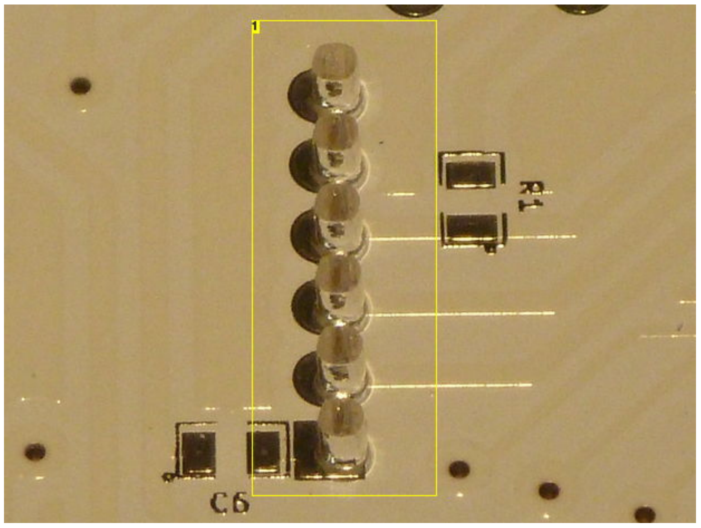


Image Notes
1. The Serial debug holes are slightly off center to stop the pins poking through the holes.

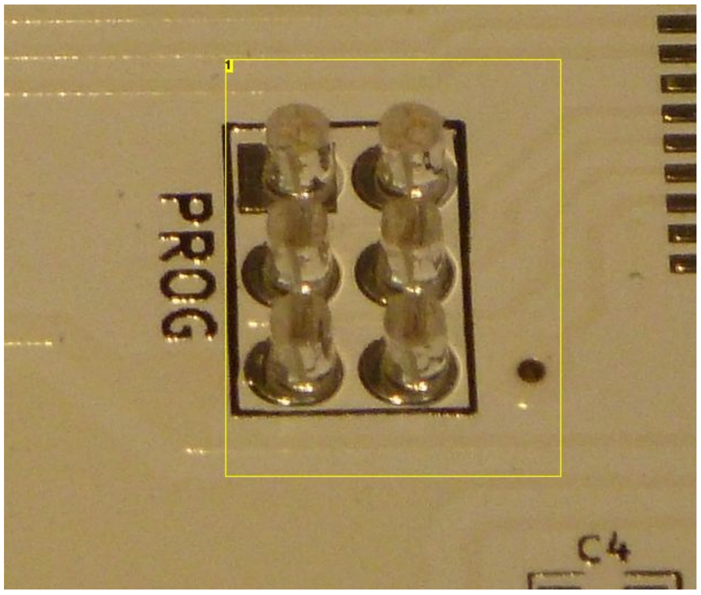


Image Notes
1. The Programming pins are also slightly off centre

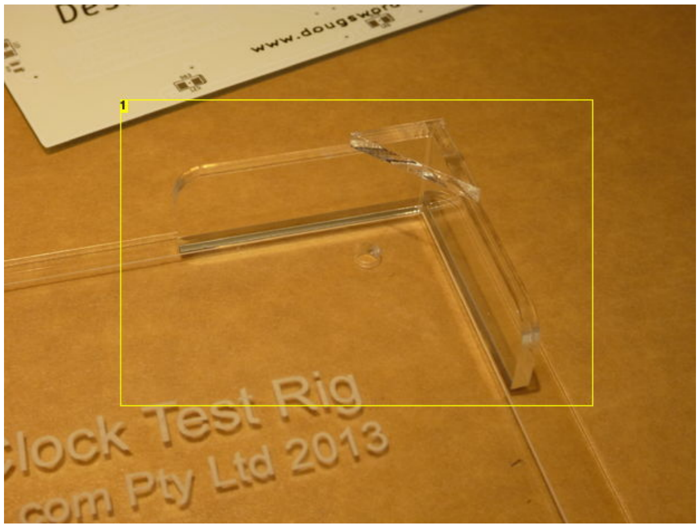


Image Notes
1. I made a corner support to keep the edge of the board aligned.

Let the adhesive cure *before* you try it out...

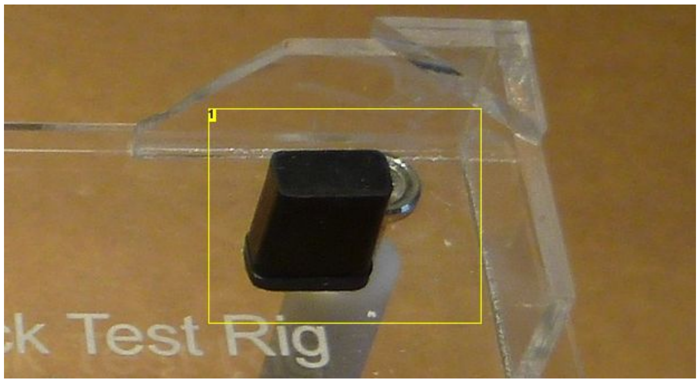


Image Notes
1. Rubber Mount - Make sure that this doesn't hit any components on the back of the board.

Step 4: A Raspberry Pi for the Brains and a 1.8" Color Display

I needed something to replace my laptop, so I decided to use the Raspberry PI.

It mounted easily at the base of the programmer and uses a simple 26 way cable connected to the GPIO pins to connect with the DeskClock board and the Display, and the toggle switch.

the specific pin configuration that I used is not important - you will use your own based on your needs.

The display that used is a 1.8 " display from Sainsmart - I brought a heap of them 6 months ago in case I found a use for them - This was just the use! I followed Marks blog http://marks-space.com/2012/11/23/raspberrypi-tft/ to rebuild a Linux kernel to support the display.

Mark was right - compiling the kernel on the Pi was a SLOW process - I left it run overnight.

Wiring up the display was simple, and quite rapidly I had a working FrameBuffer2 device.

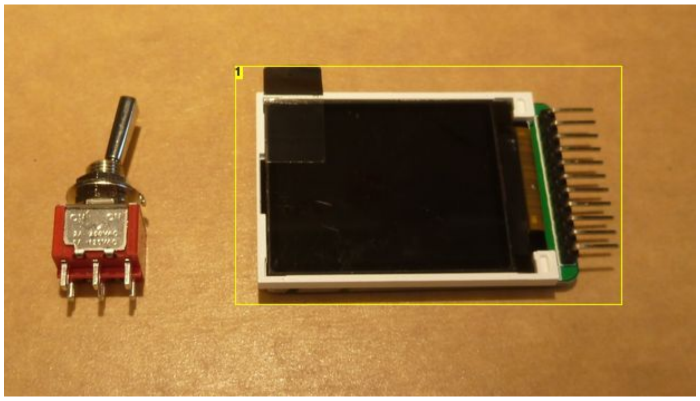


Image Notes
1. Elcheapo SainSmart 1.8" display from eBay

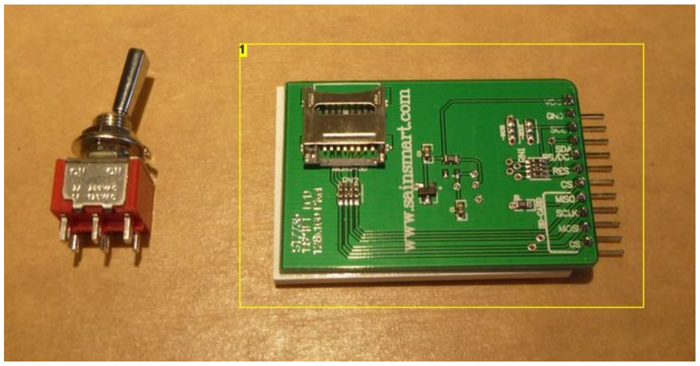


Image Notes
1. Rear of the elCheapo Sainsmart 1.8" display - We don't bother with the MicroSD card interface

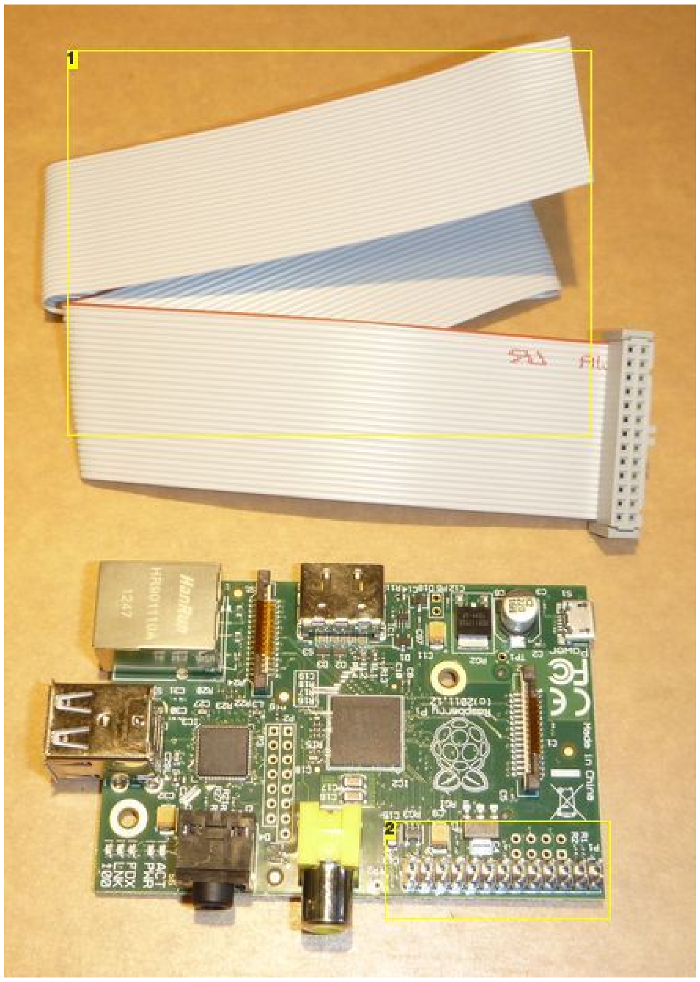


Image Notes
1. A 26 way ribbon cable for the GPIO connector
2. The GPIO connector

Image Notes
1. mounted inside the programming Jig

Step 5: A Pocket to Hold the 1.8" Display

I needed a way to mount the LCD display on the jig so that it didn't rattle around. I cam up with a simple ides - Just build an angled pocket for it.

It sits neatly at the front of the unit, at an angle for the user to easily see the display.

The display fits firmly, but in case it should decide to slide out, and 3mm nylon screw holds in in place.

It's funny, I forgot for about 20 years how easy Acrylic is to work with. I used it in shop at school, then promptly forgot about it. now, my workshop has buckets of the stuff :-)

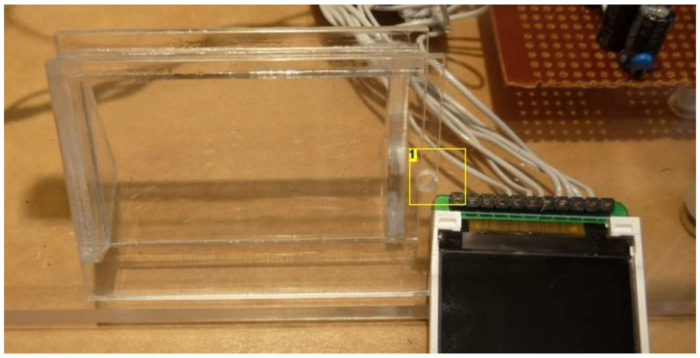
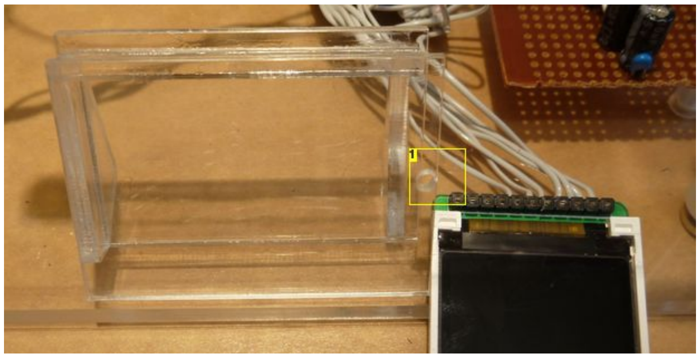


Image Notes
1. 3mm screw hole



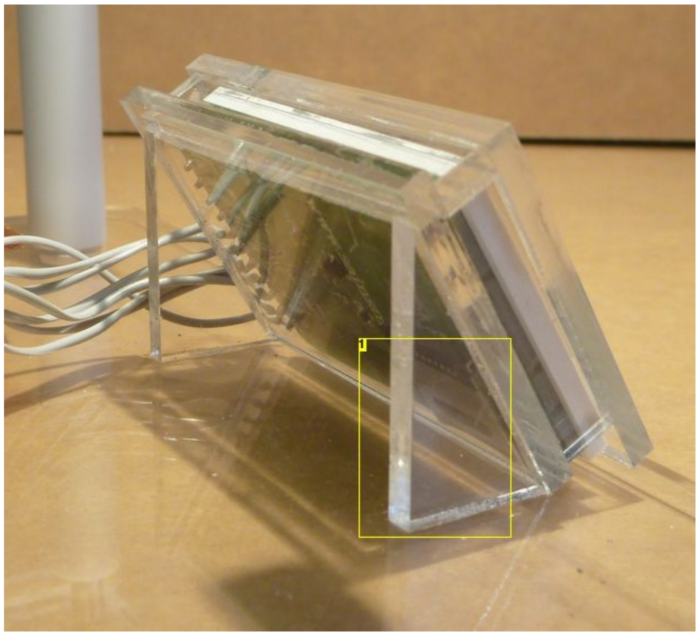


Image Notes
1. Wedge shaped supports

Step 6: Making the Pi be a Programmer

The next part of the build was finding software to allow me to program the board directly with the Pi. I decided to use the method that Steve Marple explained on his blog: http://blog.stevemarple.co.uk/2013/03/how-to-use-gpio-version-of-avrdude-on.html .

In my case, I used different GPIO pins as the 1.8" LCD display conflicted with them.

There was a loud squeal of glee when I discovered that the Pi was programming correctly.

Some people have used level shifters to protect the PI - I didn't and the project just works.

Step 7: A Toggle Switch to Remove Power from the DeskClock Board

I decided to install a toggle switch to remove power from the DeskClock board, and to tell the PI when it was time to start.

The switch was a DPDT, so one half connected to +5v and the other half connected to an unused GPIO pin.

When the switch was off, the GPIO pin was grounded, and when it was on, the GPIO pin was pulled high. i used a 100 ohm resistor to ensure that the GPIO pin was buffered in case it was set to be an output.

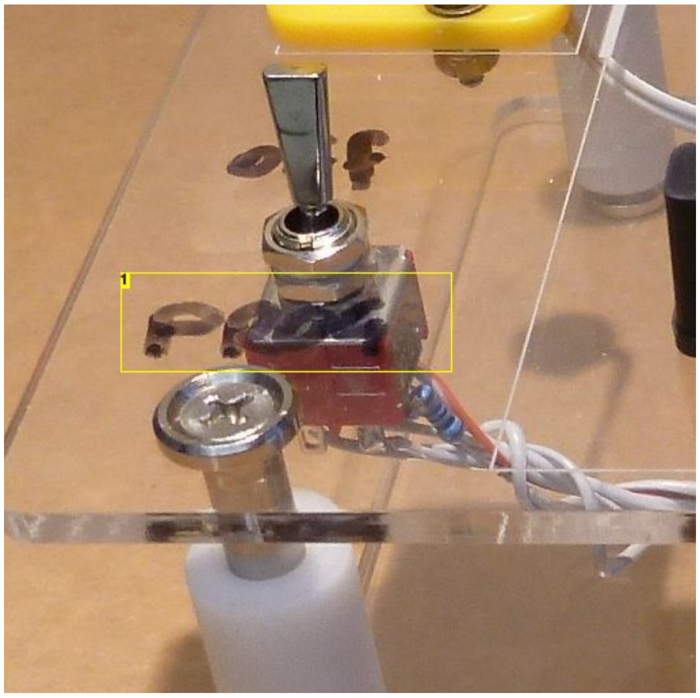


Image Notes
1. This was a late addition - I have to laser etch the words beautifully - one rainy day :-)

Step 8: Software In The Pi To Tie It All Togehter

Next, I wrote my first Python program.

I am a C programmer - Fortunately, there are buckets of tutorials to help out.

I got most of the code from a set of samples where somebody used their PI as a weather display.

Here is the code for the Python script that reads the button and controls the display

```
#!/usr/bin/python

import pygame
import sys
import time
from time import strftime
import os
import subprocess
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)



#set the framebuffer device to the TFT
if not os.getenv('SDL_FBDEV'):
os.putenv('SDL_FBDEV', '/dev/fb1')
os.putenv('SDL_VIDEODRIVER', 'fbcon')

def displayTime():
# used to display the date and time to the TFT
screen.fill((0,0,0))
font = pygame.font.Font(None,50)
now=time.localtime()

for setting in [("%H:%M:%S",60),("%d %b",10)] :
timeformat,dim=setting
currentTimeLine = strftime(timeformat, now)
text = font.render(currentTimeLine, 0, (0,250,150))
Surf = pygame.transform.rotate(text,-90)
screen.blit(Surf,(dim,20))


def displayText(text, size, line, color, clearScreen):
# used to display text on the TFT screen
if clearScreen:
screen.fill((0,0,0))

font = pygame.font.Font(None, size)
text = font.render(text, 0, color)
textRotated = pygame.transform.rotate(text, -90)
textpos = textRotated.get_rect()
textpos.centery = 80
if line == 1:
textpos.centerx = 90
screen.blit(textRotated,textpos)
elif line == 2:
textpos.centerx = 40
screen.blit(textRotated,textpos)


def main():
global screen
pygame.init()

size = width, height = 128, 160
black = 0,0,0
RED = 255,0,0
GREEN = 0,255,0
BLUE = 0,0,255
WHITE = 255,255,255

fail_cnt=0

GPIO.setup(18,GPIO.IN)


pygame.mouse.set_visible(0)
screen = pygame.display.set_mode(size)

displayText("DougsWordClock ", 20, 1, GREEN, True)
displayText("150mm Programer ", 20, 2, BLUE, False )
pygame.display.flip()
time.sleep(5)

displayText("Firmware Rev ", 20, 1, RED, True)
displayText("20130520 ", 40, 2, WHITE, False )
```

```
pygame.display.flip()
time.sleep(5)


while True:
displayText("Waiting ", 30, 1, GREEN, True)
displayText("Insert Board ", 20, 2, BLUE, False)
pygame.display.flip()
if (GPIO.input(18)):

displayText("Programming ", 30, 1, (200,200,1), True)
displayText("Wait 10 Sec ", 30, 2, RED, False)
pygame.display.flip()
```

And here is the shell script that actually does the programming:

```
#!/bin/sh
cd /home/pi
sudo avrdude -c gpio -p m169 -Uefuse:w:0xf5:m -U hfuse:w:0xDa:m -U lfuse:w:0xFF:m -Uflash:w:DeskClock-Prod.hex
```

Of course your jig will have different software :-)

Step 9: IT ALL WORKS!!!

Finally, I had the lot connected, and it worked a treat!

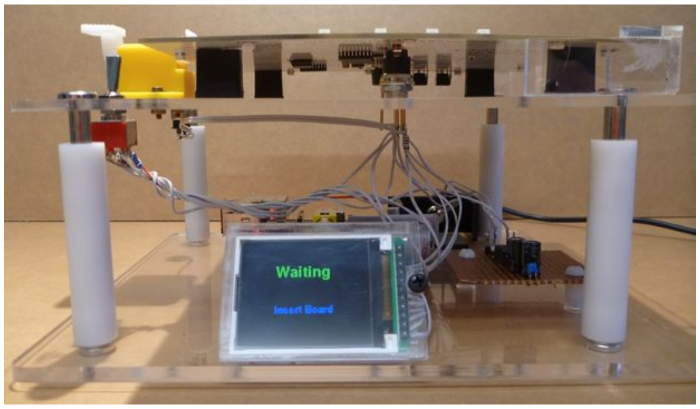
I learnt a heap about driving these little 1.8" LCD displays, to the point where they are now my turn to device for trivial Pi projects.

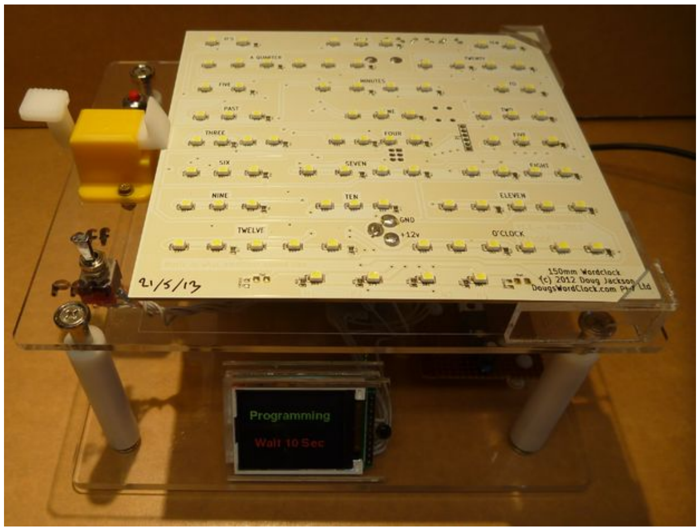
Anyway - Here are a couple of photos of it in action.

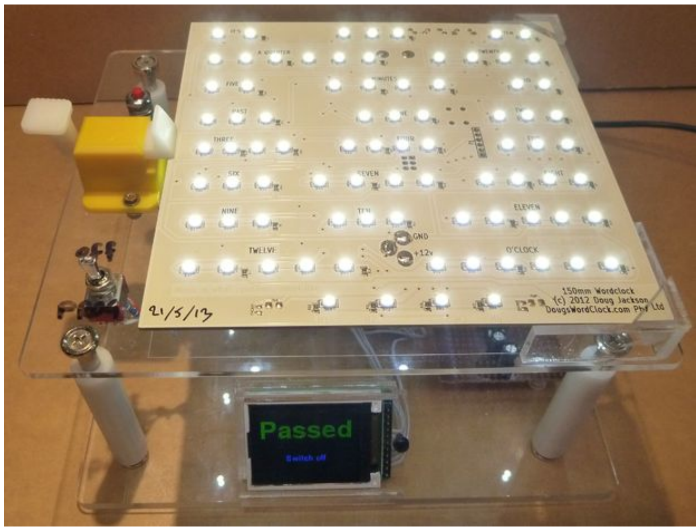
Enjoy.

Where to from here?

Well that's a cool question - At the moment, the programmer simply programs the board and verifies that the micro was flashed correctly. We inspect the operation of the LEDs visually (hence the bright display) - The next step is to add a function that can communicate with the running board to validate the accuracy of the RTC chip / crystal combination, comparing the passing of time with an internet standard. That should not be too hard..... :-)





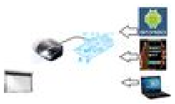


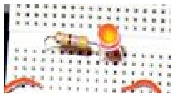
Related Instructables

0-5V Analog input from Raspberry Pi graphed on Web by electronicshouse

Autonomous, Cardboard, Rasberry Pi Controlled QuadCopter by kayak0806

Raspberry Pi as webserver. by antares72


Connect Raspberry Pi to Projector or TV by tim.ding


LED Blinking with Raspberry Pi by rahulkar

Use ssh to talk with your Raspberry Pi. by antares72

Comments

7 comments Add Comment

george graves says: Jun 10, 2013. 5:28 PM REPLY
Do you have a source for the board clamp?

drj113 says: Jun 10, 2013. 6:53 PM REPLY
I got it with this set of parts: http://www.ebay.co.uk/itm/Prototype-Test-Fixture-Jig-for-Pogo-Pin-PCB-Board-DIY-/260977323611?pt=LH_DefaultDomain_0&hash=item3cc375ee5b

george graves says: Jun 10, 2013. 7:56 PM REPLY
Thank you very much.

bulb66 says: Jun 1, 2013. 12:04 PM REPLY
Hi Doug,

Nice project, any chance you are going to release the files for the new deskclock board? Looks like a great upgrade for my Ardunio word clock :-)

J

drj113 says: Jun 2, 2013. 3:56 PM REPLY
Hi,

I have wrestled with that for a while, and because the project is 100% surface mount, I have decided not to release the files as the last thing I want is for people to be disapointed. I feel that releasing a surface mount project, while amazing for 10% of the people, is essentially setting the 90% up for a massive failure.

If you are comfortable with doing the soldering, then chuck me an email and I will send you a board so you can have a play. doug@dougswordclock.com

HelmutHound says: May 31, 2013. 10:58 PM **REPLY**

Very nice! The blogs are interesting! Any pointers you can give for a Linux beginner?

And I heard a funny rumor about the RasPi that I am hoping you could answer!
I heard that when data is written and accessed repeatedly on a RasPi, that the flash memory system corrodes and in 1 to 2 months, it will need to be replaced / repaired.
(Though less use increases this amount of time before failure)
Is this true?

drj113 says: Jun 1, 2013. 2:26 AM **REPLY**

Hi,

The best way of getting good at using Linux is to do just that - Spend the effort to be able to use it instead of your regular system.

Flash memory always has a finite lifespan - The amount of life depends on how many writes you subject it to. It generally has a life of some millions of writes - write to it 1000 times per second and it will only last 1000 seconds. That said, there are 'write levelers' and caches that reduce the actual number of writes that are experienced.