PasswordPump

User's Guide

Contents

 last revision date: 2020-07-02

 last revision date: 2020-07-02

Please Read this Before Purchasing

Before you purchase a PasswordPump it's best to make sure that you can set up and successfully run the PasswordPumpGUI, that's the Python based user interface that can be used to edit the credentials stored on the PasswordPump device. Go to the Setting Up PasswordPump GUI section of this document, follow the instructions, and confirm that you can run the user interface before you spend money on a PasswordPump. Naturally you won't be able to connect to the PasswordPump device over USB, but you'll at least know that you can run the UI.

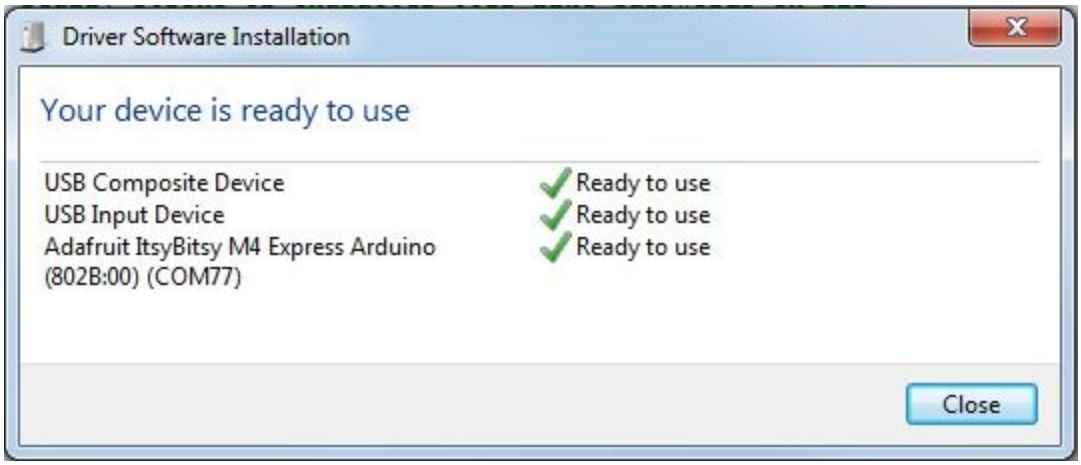
I have been using the PasswordPump for over a year now. It saves me a lot of time and aggravation and I feel way more secure about how I'm managing my accounts; especially my financial accounts. I have 133 accounts loaded on mine and almost every account in the device has a random 31 character password that I don't even know. Some folks say that if you know what all your passwords are, you're doing it wrong. The only passwords that I do know are the passwords to my Windows active directory account at work, the master password for the PasswordPump, and the password for the encrypted thumb drive on which I store my PasswordPump backups. Oh, and I know my ATM PIN.

Like most people I used the same password almost everywhere, or some variation of it. This is an extremely dangerous practice, because if hackers compromise the credentials for one of your accounts, you can bet that they will try to login to hundreds of other services using the same credentials. This is called password replay or credential stuffing. Next to phishing this is the most common method by which account security is compromised. I also keep the secondary EEprom device on the PasswordPump backed up, occasionally backup to a third EEprom device, and I religiously backup all of my credentials to a PasswordPump csv file, which I encrypt, and, in turn, store on an encrypted flash drive which, in turn, I store in a safe. This is important, because if the PasswordPump fails I don't want you to lose access to your accounts! I have worked hard to eliminate defects from the device but it's not perfect yet and it probably never will be. There are always defects in software, and the defects I'm aware of and working on are enumerated here. But it's likely there are more among the 7,800+ lines of code I've written for the project. Finally, I want you to be happy with the PasswordPump; so if you're not, let me know. -Dan Murphy

Initial Setup

If you purchased an unassembled kit go to the Assembly Instructions section of this document and assemble your PasswordPump. Please come back here when you're done.

You may notice that when you first plug in the PasswordPump that your operating system looks for drivers to install for the device. On Windows 7 and 10 I've had mixed experiences. Sometimes everything works fine without having to take additional steps (typically with Windows 10), and sometimes I have to manually install drivers supplied by AdaFruit (typically with Windows 7). I've included those drivers in the repository for the PasswordPump, here: https://github.com/seawarrior181/PasswordPump_II. Download and run adafruit_drivers_2.4.0.0.exe if necessary. Install the drivers that are selected by default. If you install the drivers (e.g. for Windows 7), in the Device Manager you'll see *ItsyBitsy M4* in the description for the device (under Ports (COM & LPT)). If you don't install the drivers (e.g. with Windows 10) you'll see *USB Serial Device* in the description for the device. This is what I see when I plug the PasswordPump into a Windows 10 computer for the first time and the drivers are automatically installed:



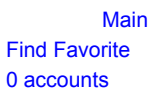
On the PasswordPump itself, when you first plug the PasswordPump into a USB port, you should see the following:

```
PasswordPump    v2.0.3
March 11 2020
(c)2020 Dan Murphy
```

At this point you want to decide on a master password. A master password should be something that you can enter reasonably quickly using the rotary encoder, so if you're going to

use a word think of one that's made up of characters from the beginning of the alphabet. For example; *cabbages* or *Abacus*. There are many other examples. You want a word or a combination of words, numbers and symbols that are not tedious to enter via the encoder. So I typically select a word that I can enter quickly followed by a four digit number. Of course you can enter anything you like, as long as it doesn't exceed 15 characters. It is possible to change the master password once you've entered it. If you want to change it and you haven't entered any credentials that you don't want to re-enter, then simply choose Factory Reset from the main menu. If you want to preserve all entered credentials, navigate to *Settings* and then to *Change Master Psswrd*.

Ok, so you've thought of a master password you want to use. To start the process <ShortClick> the rotary encoder (press it down and release it without holding it down for more than a half second). Then use the rotary encoder to scroll to the first letter you want and <ShortClick> again. Continue entering characters in this fashion until they are all entered, then <LongClick> (press the button down on the rotary encoder for more than half a second, then release it), and you'll see the following:



Remember your master password. If you forget it you'll lose access to all of the credentials you've entered, and short of breaking AES-256 encryption somehow, you're not getting them back (unless you have exercised the highly recommended feature that allows you to export all of your credentials to a file, which you should also encrypt).

Now you're ready to start entering credentials. The easiest and best way to do that is via PasswordPumpGUI (a.k.a. PassPumpGUI_v2_0.py or similar). It's a python program that serves up a user interface that you use to maintain credentials (account names, user names, passwords, previous password, URL, and credential groups). See Setting up PasswordPumpGUI for instructions on how to set up PasswordPumpGUI. To obtain the program point your browser to https://github.com/seawarrior181/PasswordPump_II, and browse to the subfolder that matches the version of the firmware that's on your PasswordPump. You can see this version number when you first power up the PasswordPump. Download PassPumpGUI_v2_0.py (or similar) and place it into the folder where you want it to reside. Also download PasswordPumpGUI.bat and place it on your desktop. Edit that file to point to the downloaded version of PassPumpGUI. Make sure you have Python 3.8, tendo, and PyCmdMessenger installed, as per the instructions. On your PasswordPump use the rotary encoder to scroll down to *Edit with Computer* and <ShortClick>. Then you need to run PasswordPumpGUI.bat on your computer by double clicking on the desktop icon PasswordPummpGUI, and it will launch PassPumpGUI_v2_0.py (or similar). Now open the correct port and start adding new sets of credentials! Again, see the instructions included herein for Setting up PasswordPumpGUI.

 last revision date: 2020-07-02

To help you navigate through the menus on the PasswordPump a map is included here. Detailed instructions for each feature with and without the PasswordPumpGUI are documented here.

Troubleshooting the PasswordPump

1) *When I run PasswordPumpGUI.py, select the correct port and click Open, I see the following error message:*

```
C:\Users\someusername\Desktop>c:\python3\python
c:\PathToPasswordPumpGUI\PasswordPumpGUI.py
COM69: Adafruit ItsyBitsy M4 Express Arduino (802B:00) (COM69)
Connecting to arduino on COM69... done.
None
Exception encountered reading return value from pyReadHead; 'NoneType' object is
 not subscriptable
None
TypeError encountered in clickedOpen(); 'NoneType' object is not subscriptable
Opened port
```

   This happens when you neglect to enter 'Edit with Computer' mode on the PasswordPump device before opening the port. If you have entered 'Edit with Computer' mode on the device and you're still seeing this error message, try power cycling the PasswordPump, restarting the PasswordPumpGUI.py program, and trying again. In the extreme situation it's necessary to restart the computer to fix issues with the port in order to resolve this problem. I have also seen this kind of behavior when the PasswordPump is connected to the computer via a USB hub instead of being plugged directly into the computer. Try to find a way to plug the PasswordPump directly into the computer bypassing the USB hub.

2) *When I select File->Import from PasswordPump from PasswordPumpGUI.py, navigate to and select a file, some sets of credentials import but eventually I see the following error message and the GUI freezes:*

```
Error encountered reading file in ImportFilePasswordPump; 'NoneType' object is not
subscriptable
```

   Click on the Exit button. If the UI remains frozen, click on the [x] close icon in the top right hand side of the window (under Windows). On the PasswordPump, long click the rotary encoder. If that doesn't return control, click on the reset button on the PasswordPump. Login to the device and re-select 'Edit with Computer'. Make sure that,

in the export file, no fields contain a | (pipe) or a ~ (tilde). Make sure that none of the URLs end in / (forward slash). Launch PasswordPumpGUI.py and, once you select and open the correct port, re-try the import operation.

3) *When I try to login to an account using the PasswordPump, I'm told that my credentials are wrong.*

   Make sure that the caps lock key on your keyboard isn't on. If it is, everything entered by the PasswordPump comes through to the UI in upper case, butchering your password..

4) *When I change an existing account name a new account is created with the new account name and no attributes populated, and the old account remains.*

   There's presently no easy way to change the account name. The best way to rename an account is to insert a new account with the desired account name and attributes, and then delete the old account. If you edit the account name of an existing set of credentials, a new account is created with the new account name when focus leaves the account name field, and all of the other attributes are initialized to be empty. The previous account remains (you may delete it if you like). I've not decided yet if this is the desired long term behavior.

5) *When I enter an account name, username, password, or any other field that contains a ~ (tilde), a | (pipe), a " (double quote), or a , (comma) that character is changed to a # (hashtag). This behavior is also observed when importing data with tildes, pipes, double quotes and/or commas.*

   Tildes, pipes, double quotes and commas are not supported in any of the fields. You'll need to eliminate them from your account name, username, password or other fields. If you enter them from the PasswordPumpGUI they are automatically changed to # (hashtag). It's not possible to add them via the rotary encoder (unless you are entering credentials via the keyboard and a serial monitor, which is unusual).

6) *When navigating between accounts via the PasswordPumpGUI program the fields get out of sync; for example the account name appears on the Username text box, or fields are otherwise out of sync.*

   The most likely cause of this problem is a | (pipe) character embedded in the account name, username, password, old password, or URL fields. To find the offending field edit the account via the rotary encoder on the PasswordPump device. On the PasswordPumpGUI you can navigate between accounts by using the Next button, and when the alignment of fields looks wrong, take note of the previous account visited. The

problem is most likely in one of the fields of the previously visited account. You can open up Notepad (if you're running Windows), and paste the username, password, account name, URL and old password into Notepad (using the PasswordPump device). Note if any of the fields have an embedded | (pipe) character.

7) *After clicking on or navigating to an account via PasswordPumpGUI, the following error message is displayed in the python console:*

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "c:\python3\lib\tkinter\__init__.py", line 1883, in __call__
    return self.func(*args)
  File "c:\repos\murphyrepo\dev\python\PassPumpGUI\PassPumpGUI_v0_7.py", line 50
9, in clickedNext
    OnEntryDownNoEvent()
  File "c:\repos\murphyrepo\dev\python\PassPumpGUI\PassPumpGUI_v0_7.py", line 56
8, in OnEntryDownNoEvent
    OnEntryDown(0)
  File "c:\repos\murphyrepo\dev\python\PassPumpGUI\PassPumpGUI_v0_7.py", line 57
8, in OnEntryDown
    clickedLoad()                                                    # cal
ls getRecord()
  File "c:\repos\murphyrepo\dev\python\PassPumpGUI\PassPumpGUI_v0_7.py", line 63
4, in clickedLoad
    getRecord()
  File "c:\repos\murphyrepo\dev\python\PassPumpGUI\PassPumpGUI_v0_7.py", line 70
3, in getRecord
    response = c.receive()
  File "c:\repos\murphyrepo\dev\python\PassPumpGUI\PyCmdMessenger\PyCmdMessenger
.py", line 280, in receive
    raise ValueError(err)
ValueError: Number of argument formats must match the number of recieved argumen
Ts.
```

This happens when there are corrupt values in the old password field. To fix it, simply set focus in the Old Password text box and set focus on the account or password text box to save an empty string to old password on the PasswordPump. Do not hit tab after clicking in the Old Password text box or you might corrupt the value in the URL field. I believe the defect that created this situation is addressed so if you encounter this problem please report it to me.

8) *I entered a duplicate account name and lost all of my credentials for all accounts.*

While not possible solely via the PasswordPumpGUI, it is possible to enter a duplicate account name via the rotary encoder on the PasswordPump or via a combination of the PasswordPump and the PasswordPumpGUI. When you delete one or both of these accounts the PasswordPump can become corrupt, so it's important to have a backup of

all of your sets of credentials so that you can restore back to a known good state. This is an open defect that I am trying to reproduce so that I can work to address it.

9) *During import of a large PasswordPump format file, the process stops with an error.*

   There is a defect in the PasswordPumpGUI whereby an error is occasionally encountered during the import of a (typically) large PasswordPump formatted file. This problem is intermittent, and therefore difficult to pin down. For now the best approach to dealing with it is to just start from the beginning; i.e. factory reset your device and re-initiate the import operation. In terms of frequency, I estimate that for every account you import there is approximately a 1 in 300 chance of encountering the error. If you encounter this problem with greater frequency please contact me.

10) *When I press on the screen hard enough the PasswordPump resets itself.*

    The reset button on the ItsyBitsy M4 is located under the screen, so if you press on the screen hard enough you'll actuate the reset button. Don't do that. To reset the device use the button on the bottom of the device instead.

11) *A certain field of a certain account will not, under any circumstances, store a particular value for that field. For example, I am trying to set my Password to abcdefg (you would never really do that because that's a lousy password but this is just an example...). I set focus to the Password field, enter abcdefg. When I return to that account the Password field is blank. If I put any other value in the field; e.g. abcdefgh, or abcdef, this freaky behavior doesn't happen. This is annoying, what's going on?*

    This happens under certain rare circumstances and is related to how we encrypt and decrypt passwords (and all other fields in the account, for that matter, with the exception of style and group). The solution to this problem is tedious; you should either change the value that you're trying to store in that field, or you should delete the account and re-insert it. This problem is more of an annoyance when you're importing a large number of credential sets, because there's no way to know if a certain field on a certain account was dropped. Fortunately it doesn't happen very often.

12) *On one of my account names, the saved account name is shorter than that which I entered, and I've entered less than 31 characters.*

    Account name can be up to 31 characters long. However, sometimes they are truncated even further. This is a cousin to the problem above. It doesn't happen very often, but it can happen. The workaround is to either accept the shortened name, or to change the account name altogether. Remember that the account name isn't the username, it's not typically supplied when you're authenticating, so you can make it whatever you want.

This problem only affects the account name field.

13) *When my account name has commas in it, if I visit the account name field in the PasswordPumpGUI, after I reload the accounts (exit and restart PasswordPumpGUI), the commas are replaced with hashtags and all of the other fields are blank.*

    Don't import credentials with commas. If you have an account with a comma do not set focus on the account field in the PasswordPumpGUI. A fix is underway.

Description of the PasswordPump

This is v2.0 of the PasswordPump, a USB device that manages credentials for up to 250 accounts. Credentials (account names, usernames, passwords, URLs and old passwords) are stored ONLY on the device itself, on two removable EEprom chips using military grade encryption (AES-256). The credentials are not stored in the cloud or in a file on your computer where they are more exposed to hackers. Credentials are backed up on the device itself; i.e. encrypted credentials are moved from the primary EEprom chip to the backup EEprom on demand. You may remove the EEprom chips from the device (perhaps to keep a third or fourth backup). Credentials are entered either via the rotary encoder (on the left), via keyboard and serial terminal, or via a Python based graphical user interface (the PasswordPumpGUI). The device itself is approximately 1 1/8 x 2 3/4 inches, or 29 x 71 millimeters. Currently it's not housed inside of a case, but it should be and will be once a design of the case is complete. If you design a case for the PasswordPump please share it with us!

Features

(bolded items are new PasswordPump v2.0 features)

- Stores up to 250 sets of credentials.
- Authenticates with a 15 character master password.
- Search for accounts.
- Data entry via rotary encoder or keyboard and serial monitor, or via client **Python GUI** running in Windows, Ubuntu, or MacOS.
- Sends a username and password to a computer as if typed in via the keyboard. Can also send **URL**, **old password** and account name.
- Add account name, username, password (generated or not), **URL**, **old password**
- Accounts are added in alphabetical order.
- Delete an account.
- Edit existing username, password, URL, style (inter-username/password character, <Return> or <Tab>), **old password, credential groups.**
- Generate 31 character random passwords from the PasswordPump **or via the client GUI.**

- **Automatically saves the old password if it's not already populated when you generate a password.**
- Backup all accounts to a second encrypted external EEprom.
- Logout / de-authenticate via the menu, **automatically locks the computer.**
- Configurable password display on or off.
- **Configurable** failed login count factory reset (**3, 5, 10 or 25**).
- **Configurable** automatic logout after count of minutes (**30, 60, 90, 120, 240, 1 or Never**).
- **Configurable** RGB LED intensity (**high, medium, low or off**).
- All account names, usernames, passwords and URLs are encrypted w/ **AES-256**.
- Master password is hashed w/ SHA-256.
- All encrypted credentials fields and the hashed master password are salted.
- The device is not vulnerable to standard password attacks. See disclaimers.
- **The master password can be changed.**
- **Export to PasswordPump formatted CSV file.**
- **Import from PasswordPump formatted CSV file.**
- **Import credentials from Chrome export.**
- **Import credentials from KeePass export.**
- **Associate credentials with custom groups for better organization; search by group (defaults are Favorites, Work, Personal, Home, School, Financial, Mail or Health).**
- Decoy password feature that automatically factory resets the device if entered (e.g. while the user is under duress).
- **Pre-auto-logout indicator/countdown via red and blue flashing RGB LED.**
- Factory reset via menu (when authenticated) wipes out all credentials.

Disclaimers

- The PasswordPump is not secure from keylogging attacks (https://en.wikipedia.org/wiki/Keystroke_logging). Keylogging attacks are capable of stealing passwords that are entered through your keyboard. All data sent to your computer with the PasswordPump enters the computer as if through the keyboard. Therefore you should remain diligent about protecting yourself from these kinds of attacks. See the countermeasures section of the Wikipedia link provided above.

- The contents of the EEprom chips on the PasswordPump are encrypted with AES-256, and the master password is hashed with SHA-256. The unhashed master password serves as the encryption key (along with 16 bytes of salt). The credentials are also salted. Nevertheless, if somebody with nefarious purposes obtains access to your PasswordPump it's best to assume that all of your credentials have been compromised. It is possible to move the encrypted contents of the EEprom chips to an operating system file by using a USB programmer (e.g. TL866II Plus). This would allow an attacker to circumvent the protections built into the device that prevent more than 3, 5, 10 or 25 failed login attempts before the credentials are wiped. I will consider removing this advice when the device's software has been subjected to a rigorous code review by encryption industry experts.

- Under no circumstances and under no legal theory, whether in tort (including negligence), contract, or otherwise, shall the creator of this device and software be liable to any person for any direct, indirect, special, incidental, or consequential damages of any character arising as a result of the use of the PasswordPump including, without limitation, damages for loss of goodwill, work stoppage, computer failure or malfunction, personal injury, death or any and all other damages or losses.

- **This program and device are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.**

Menu Navigation on the PasswordPump

You move through the menu items by turning the rotary encoder, clockwise to move down the list and counter clockwise to move up. Account names are stored in alphabetical order. To

select an item you click down on the rotary encoder (short click). To backup you hold the rotary encoder down for more than a half second (long click).

- Master Password
- Find Favorite
- Find All Accounts
  - [scroll through accounts list]
    - Send Password <RET>
    - Send User & Pass
    - Send URL
    - Send User Name
    - Send Pass (no <RET>)
    - Send Account
    - Edit Credentials
      - Edit Account Name
      - Edit User Name
      - Edit Password
      - Edit URL
      - Indicate Style
      - Assign Groups
        - Favorites
        - Work
        - Personal
        - Home
        - School
        - Financial
        - Mail
        - Health
      - GeneratePassword
      - Save to Old Password
    - Delete Credentials [confirm]
    - Send Old Password
- Find By Group
  - Favorites
    - [same as under Find All Accounts]
  - Work
    - [same as under Find All Accounts]
  - Personal
    - [same as under Find All Accounts]
  - Home
    - [same as under Find All Accounts]
  - School

- [same as under Find All Accounts]
- Financial
  - [same as under Find All Accounts]
- Mail
  - [same as under Find All Accounts]
- Health
  - [same as under Find All Accounts]

Add Account
- Account Name
- Edit User Name
- Edit Password
- Indicate Style
- GeneratePasswrd

Logout & Lock

Backup/Restore
- Backup EEprom [confirm]
- Restore EEprm Backup [confirm]

Settings
- Keyboard ON/OFF
- Show Password ON/OFF
- Decoy Password ON/OFF
- RGB LED Intensity
  - High
  - Medium
  - Low
  - Off
- Timeout Minutes
  - 30
  - 60
  - 90
  - 120
  - 240
  - Never
  - 1
- Login Attempts
  - 3
  - 5
  - 10
  - 25
- Rename Groups
  - Edit Group 1
  - Edit Group 2

Edit Group 3
Edit Group 4
Edit Group 5
Edit Group 6
Edit Group 7
Change Master Psswrd
Factory Reset [confirm]

Operation of the PasswordPump via Rotary Encoder

To turn the device on you simply plug it into a USB port/receptacle using a USB Micro-B plug to USB-A plug cable, the same cable that you'd use to charge an Android phone. The first time you plug it in a driver **might** need to be installed. The driver is available for download in the source code repository here: https://github.com/seawarrior181/PasswordPump_II. If the device was shipped to you, assembled or as a kit, it arrives already flashed with the PasswordPump program.

The first time you power the device on you'll see :

```
PasswordPump    v2.0.3
April 23 2020
(c)2020 Dan Murphy
```

At this point you'll want to enter your master password. Try to select a password that can be more quickly entered into the device. It should be a combination of upper and lower case, with numbers and maybe a symbol or two. I like to pick a password that can be typed almost entirely with my left hand, I find they are easier to input via the rotary encoder. You should select a strong password; a combination of letters, upper and lower case, numbers, and special characters, between 7 and 15 characters long. To enter a character turn the rotary encoder until the character appears and then press the rotary encoder down (short click) to select the character. There's presently no way to back up if you make a mistake so be careful. Once the entire master password has been entered long click the device (click down the rotary encoder for more than 1/2 a second). You've just entered the master password and now you're ready to enter a set of credentials. Don't forget your master password, it's the only way to recover your encrypted credentials short of cracking SHA-256 or AES-256.

You move through the menu items by turning the rotary encoder, clockwise to move down the list and counter clockwise to move up. Account names are stored in alphabetical order. To

select an item you click down on the rotary encoder (short click). To backup you hold the rotary encoder down for more than a half second (long click).

Note: The following instructions describe the easiest way to enter credentials if you don't have access to the PasswordPumpGUI or if it's not working correctly. The easiest way to enter credentials is via the PasswordPumpGUI, and it's fairly self-explanatory, so use that method if possible.

Adding Credentials via Keyboard

You can add credentials via the PasswordPump by entering them directly with the rotary encoder or by using a keyboard in combination with a serial terminal. To add a set of credentials via the keyboard you need to open a serial terminal. The one that works best for me is the Arduino serial terminal. So if you open the Arduino IDE go to Tools->Ports and select the *Adafruit ItsyBitsy M4 (SAMD51)* port. Then select Tools->Serial Monitor (or Ctrl+Shift+M). Next, on your PasswordPump navigate down to Keyboard OFF and change it to Keyboard ON with a short click. Navigate back up to Add Account and short click. You'll see:

Edit Credentials
Edit Account Name

Short click, and you will see

Account Name
Edit Account

Switch back to the Arduino Serial Terminal and enter the account name, followed by the return key. Then long click on the Password Pump. You should now see:

Edit User Name
[the account name you entered]

Short click again, switch back to the Arduino Serial Terminal and enter the username, followed by the return key. Then long click on the Password Pump. You should now see:

Edit Password
[the account name you entered]

Short click again, switch back to the Arduino Serial Terminal and enter the password, followed by the return key. Then long click on the Password Pump. You should now see:

Indicate Style

[the account name you entered]

Short click again and use the rotary encoder or the keyboard and serial terminal to specify either 0 or 1. Specify 0 if, while supplying username and password, the Password Pump should send a carriage return after sending the username and before sending the password. Specify 1 if, while supplying username and password, the Password Pump should send a tab after sending the username and before sending the password. Then long click on the PasswordPump. You should now see:

Account Name
[the account name you entered]

Long click again and you'll see:

Find Account
[the account name you entered]

You've finished entering the credentials.

Note that you can also enter credentials using just the rotary encoder. Keyboard can be ON or OFF, it doesn't matter. Simply enter the credentials using the rotary encoder in a fashion similar to how you entered the master password.

Sending Credentials

Navigate to *Find All Accounts* and short click. Use the rotary encoder to scroll through the list of credentials you've entered. When you've found the account name associated with the credentials you want to send to your computer, place the input focus in the username text box in the window prompting you for credentials on your computer. On the Password Pump you should see:

Send Password <RET>
[the account name you selected]

Scroll down one menu item with the rotary encoder and you'll see:

Send User & Password
[the account name you selected]

Short click to send the user name, a carriage return or a tab character (depending on the style setting), and then the password. If you selected the correct style you should now be logged in to your account / application.

If you only want to send the password to the computer, followed by a carriage return, scroll back up once using the rotary encoder until you see:

Send Password <RET>
[the account name you entered]

And short click to send the password and the carriage return character.

Similarly you can send just the user name or just the account name or url.

Editing Credentials

To edit a set of existing credentials first decide if you're going to edit the credentials via the keyboard or just the rotary encoder. If you're going to edit the credentials via the keyboard follow the instructions in *Toggling Keyboard Entry.* Then use *Find All Accounts* to navigate to the account you want to edit and short click. Then scroll down to *Edit Credentials* and short click. Then scroll to the attribute you want to edit; *Edit Account Name, Edit User Name, Edit Password*, *Edit URL*, or *Indicate Style*. Now short click. Use the keyboard to re-enter the attribute in the fashion described in *Adding Credentials*, or just use the rotary encoder to re-enter the attribute. Then long click to save the change. If you are generating a new password for the account then follow the instructions in *Generating a Password.*

Deleting Credentials

Make sure you have a current EEprom backed up. Navigate to *Find All Accounts* and short click. Use the rotary encoder to select the account that you want to delete, and short click. Using the rotary encoder scroll down to *Delete Credentials* and short click. Confirm your desire to delete the account by selecting *Y* with the rotary encoder and short clicking. The account is gone now and it's wiped from the primary EEprom chip. It isn't wiped from the backup EEprom yet, so if you accidentally delete an account, and you have a recent backup, you can restore the backup and the account will reappear. Navigate to *Find All Accounts* and verify that your account is deleted. If you're not able to scroll through all of your accounts, an intermittently occurring defect has occurred and the linked list that manages the display of all of the accounts is corrupted. Restore the latest backup from EEprom. If you backup the EEprom immediately after deleting the account it is also wiped from the secondary EEprom.

Generating a Password

Read through all of these instructions before attempting to change your password to a new generated password. The most powerful feature of the PasswordPump is its ability to generate random 31 character passwords and remember them. These passwords are extremely difficult to guess and are not as vulnerable to brute force attempts to break into an account. Before performing this operation you should be sure that you have a current backup of all your credentials. When you generate the new password, the existing/old password will be moved to the Old Password attribute **if it is empty.** If Old Password is not empty it will **not** be overwritten.

So you will probably want to blank out Old Password before generating the new password. To generate a password for an account simply find the account via *Find All Accounts* and select the credentials by short clicking on the account name. In your application on your computer navigate to the change password feature and place input focus in the Old Password text box. On the PasswordPump navigate to *Send Password* (NOT *Send Password <RET>*) and short click. In your application on your computer, place input focus in the new password text box (typically by hitting the <TAB> key). In the PasswordPump scroll down to *Edit Credentials* and short click, then scroll down to *Generate Password* and short click. This changes the password to a randomly generated series of 31 characters. Now long click once, navigate to *Send Password* (NOT *Send Password <RET>*) and short click. If you need to confirm the new password then place input focus on that text box in the application on your computer and short click again. Confirm your password change by hitting the return key or otherwise clicking on the appropriate button. You now have a random 31 character password on the account, and the only place where that password exists is on the encrypted EEprom chip on your PasswordPump. At this point it's a good idea to *Backup to EEprom* and *Backup to a File,* and to be sure that you can somehow recover from a lost password on that account. Warning: If the attempt to change your password fails because the existing/old password is not accepted be aware that you have just overwritten the old password with your new generated password. To access the old password you'll need to either use the Old Password attribute (assuming it was blank before you generated the new password), *Restore a Backup from EEprom* and try again, or go to the encrypted backup file on your thumb drive to get the current password for the account, or recover the password from the account using whatever mechanism is available to you via the application or web site. Think ahead and be careful so that you don't lock yourself out of your account!

Logging Out and Locking Your Computer

When you want to log out of the device navigate to *Logout & Lock* using the rotary encoder and short click. The RGB led changes from green to blue. You're now logged out of the PasswordPump and must enter the master password again in order to use the device. In addition to locking the PasswordPump, this also locks your computer so that you'll need to re-authenticate to gain access to your computer. If you want to log out of the PasswordPump without locking the computer simply press the reset button on the bottom of the PasswordPump.

Toggling Keyboard Entry

Navigate to *Settings,* single click, and navigate to *Keyboard.* Short click to toggle the setting. When the keyboard is on you may enter credentials via the keyboard and serial terminal using the process described in *Adding Credentials.* Keep the keyboard set to OFF when you're not entering credentials via a serial terminal and the keyboard. This setting is saved when the PasswordPump is powered off.

Showing/Hiding Passwords

Using the rotary encoder navigate to *Settings,* single click, then navigate to *Show Password.* Short click to toggle the setting. This setting is saved when you log out and power down the device. This setting determines if passwords are shown or hidden on the PasswordPump. The setting for the PasswordPumpGUI is independent.

Decoy Password

Using the rotary encoder navigate to *Settings,* single click, then navigate to *Decoy Password.* This setting controls behaviour whereby the PasswordPump is factory reset when you enter your password followed by the uppercase characters FR when logging into the PasswordPump. This is useful if someone is forcing you to authenticate to the PasswordPump and you want to immediately Factory Reset the device. Remember that if you enter the decoy password you will lose all of the credentials stored on the primary and secondary EEprom chips installed on the PasswordPump.

RGB LED Intensity

You can control the intensity of the RGB LED by navigating to *Settings* and selecting *RGB LED Intensity.* Select *High*, *Medium*, *Low*, or *Off* using the rotary encoder. Long click to save your setting.

Automatic PasswordPump Logout

To control the duration of PasswordPump inactivity time after which you will be automatically logged out of the PasswordPump, navigate to *Settings*, then to *Timeout minutes*, and set your inactivity time to *30, 60, 90, 120, 240, 1* or *Never*. Note that the inactivity timer on the PasswordPump does not lock your computer screen (although a sound security practice is to set a timeout on your computer for your computer, as well).

Login Attempts

To set the number of failed login attempts allowed before a factory reset of the PasswordPump is performed, navigate to *Settings* and *Login Attempts.* You can select *3, 5, 10,* or *25* failed login attempts.

Backing Up to EEprom

On the Password Pump navigate to *Backup/Restore,* then to *Backup EEprom* using the rotary encoder. Short click, then confirm that you want to copy credentials and settings from the primary EEprom to the secondary EEprom by selecting *Y* with the rotary encoder and short clicking. The RGB will be yellow while the backup is taking place, and then change back to green. It should only take about two seconds to complete this operation.

Restore a Backup from EEprom

If you decide that you want to restore the EEprom backup (or, in other words, have the contents of the secondary, backup EEprom overwrite the contents of the primary EEprom), then navigate to *Backup/Restore,* then to *Restore Backup*, on the PasswordPump. Short click and confirm the operation by selecting *Y* with the rotary encoder and short clicking. The RGB led will turn yellow until the operation is complete, then it changes back to green. The master password remains the same. This operation completes in about two seconds.

Rename Groups

Using the Rename Groups option it's possible to customize the names of the groups. By default those names are Favorites, Work, Personal, Home, School, Financial, Mail, and Health. You can change any and all of these names to suit your needs. The group names cannot exceed 10 characters.

Change Master Password

If you want to change your master password note that you can achieve this via the PasswordPumpGUI or via the rotary encoder on the PasswordPump. If you want to change the master password via the rotary encoder, navigate to *Settings* and *Change Master Psswrd.* Single click, and then carefully enter the master password via the rotary encoder. When you're done, long click and wait for the process to finish. The RGB LED will be yellow while the credentials are being backed up to the secondary EEprom (for about two seconds), and then quickly flash yellow while it's re-encrypting all of your credentials and copying them back to the primary EEprom (for about 5 seconds). Check all of your credentials after changing the master password. If you are not happy with the results you can restore the backup from EEprom (see above) and reinstate the former master password. If you are happy with the results, back up *to EEprom*. If for whatever reason you cannot remember your new master password just after changing it, swap the positions of the EEprom chips on the PasswordPump device and login with the old master password.

Performing a Factory Reset

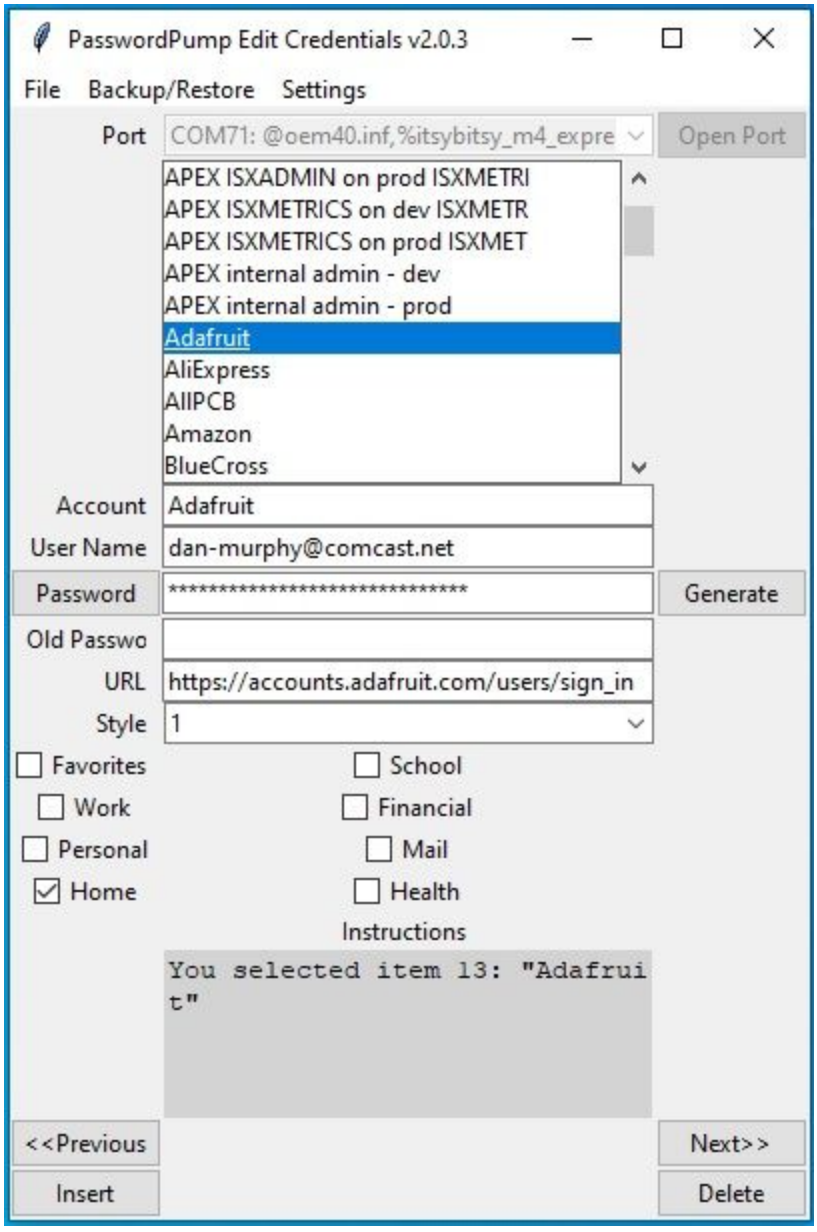
You want to wipe out all of the encrypted credentials on the primary and backup EEprom and factory reset the device. On the PasswordPump navigate all the way down to *Reset* using the rotary encoder. Short click. Confirm that you want to factory reset the device and clear all of the credentials and the master password from both EEprom chips by selecting *Y* with the rotary encoder and short clicking. The RGB will flash blue and red slow and then fast while the device is factory resetting, then change to blue. At this point you can enter a new master password. Note that a Factory Reset also wipes out the credentials stored on the backup EEprom.

Groups

Groups allow you to assign groups to credentials so that you can find them faster when you're trying to send them. The default groups are *Favorites, Work, Personal, Home, School,*

Financial, *Mail* and *Health*. These group names, except for *Favorites,* are configurable. You'll notice that the default credential search on the main menu is *Find Favorites*. After that you encounter *Find All Accounts*, and then *Find By Group*.

Setting Up PasswordPumpGUI



Download Python 3.8 for your computer's operating system from here: https://www.python.org/downloads/release/python-381/. After installing Python 3.8, use pip to install the *tendo* and *PyCmdMessenger* packages:

```
pip install tendo
pip install PyCmdMessenger
```

You may need to install Tkinter:

```
sudo apt-get install python3-tk
```

Now you can download PassPumpGUI_v2_0.py from this location: https://github.com/seawarrior181/PasswordPump_II/blob/master/v2_0_3/PassPumpGUI_v2_0.py. Save the file to your desktop. Then create PasswordPumpGUI.bat and save that to your desktop as well. Here are it's contents (assuming you're on Windows and you installed Python 3.8 to C:\python3)::

```
c:\python3\python c:\yourUsername\desktop\PassPumpGUI_v2_0.py
```

Substitute *c:\python3* from above with the location where you installed Python 3.8, and substitute *yourUsername* with your username. Now place your PasswordPump into *Edit with Computer* mode and you should be able to double click on PasswordPumpGUI.bat from your desktop to launch the PasswordPumpGUI python program. Open the correct port and you'll be able to edit credentials from the GUI.

Importing and Exporting Files with PasswordPumpGUI

One of the best features of the PasswordPumpGUI is that it allows you to import a couple of file formats and export to what I call the PasswordPump format. All of these formats are .csv files, or files full of comma separated values.

PasswordPump Format

The PasswordPump format looks like this:

```
accountname, username, password, oldpassword, url, style, group
```

For example:

```
"_Active Directory", "YOURDOMAIN\yourname", "yourpassword",
"yourlastpassword", "", "1", 75
```

Or

```
"Instructables",”yourusername","yourpassword","lastpassword","www.ins
tructables.com","1",4
```

If you import from a file that's in PassworPump format individual credential sets will need to be in the format specified above. And when you export to the PasswordPump format, which is recommended for keeping backups, it will produce a file in the format above, including the header row..

KeePass Format

The PasswordPumpGUI will also allow you to point it at files in a KeePass .csv format and move those credentials into the PasswordPump. That format is as follows:

```
Account, Login Name, Password, Web Site
“Yahoo Mail”, “myYahooName”, “q9jc34j043”, “https://login.yahoo.com”
```

Do not remove the heading row if it exists, it's necessary to process the file. If your export file contains a row for comments you will need to delete that column.

Chrome Format

The PasswordPumpGUI will also allow you to point it at files in Chrome .csv format and move those credentials into the PasswordPump device. The expected format is as follows:

```
name, url, username, password
“accounts.adafruit.com”,”https://accounts.adafruit.com/users/sign_in,
joe_customer@gmail.com”,”984hfu98qpf4qnv”
```

Tips & Tricks

- Do not make a habit out of unplugging the device from it's micro-B USB port. Instead unplug the end of the cord that plugs directly into the computer (USB A), and leave the device plugged into the cord. This reduces the wear and tear on the device's micro USB port and will extend the life of the unit. I have seen similar micro-B USB ports fail, especially on the cheap Chinese made ATMega 32u4 boards that were sometimes used

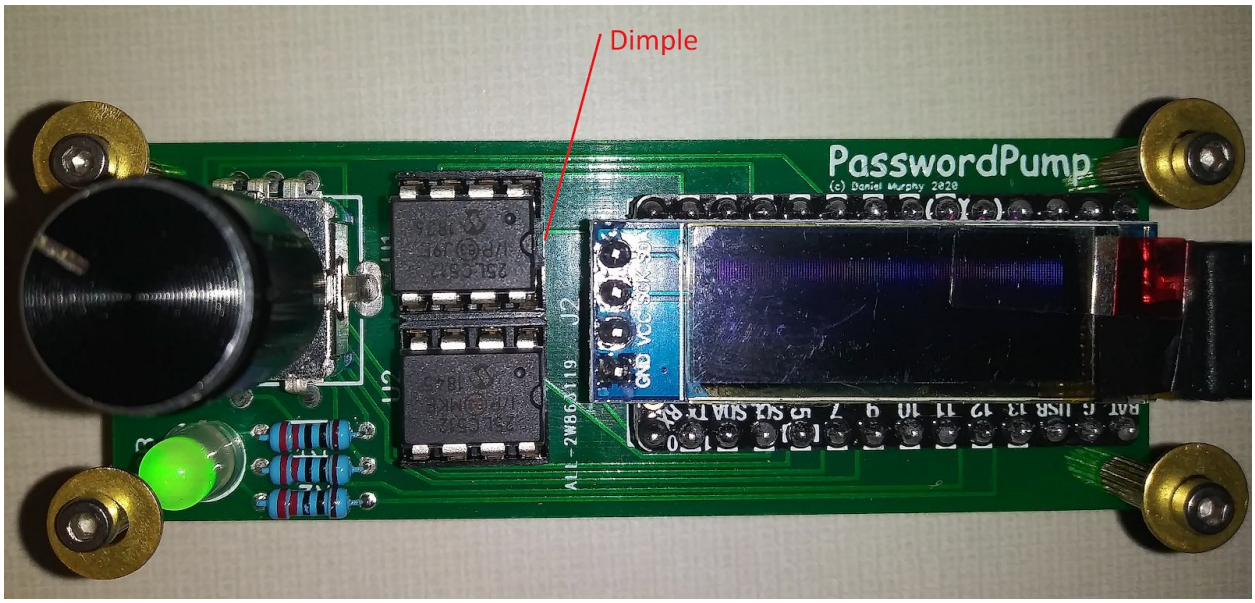
for version one of the PasswordPump. I am now recommending against the use of the magnetic USB cables, I have observed some intermittently weird behavior on Windows, Ubuntu and Raspbien when using them, specifically (on Windows), the "Unable to recognize USB device" error.

- After you create a KeePass or a Chrome export file, and before importing into the PasswordPump, edit the .csv file and make sure that none of the accounts have embedded commas (,), pipes (|), tildes (~), or backslashes(\). These characters Atend to create problems and I am working on solutions. After removing the problematic characters save the .csv file before importing. You may need to change some of your existing passwords if they contain the problematic characters.
- If you have many accounts, associate the accounts you use the most with the Favorites group. Of these favorite accounts, name the accounts you use the very most with an _ (underscore) first so that they will sort to the top of the Favorites list. I use this technique to identify my MS Active Directory credentials, which I have to supply in many places. That account is named _Active Directory, so it always sorts to the top. After I login to the device I can short click three times and my _Active Directory password is typed into the computer via the PasswordPump. Using this technique my most frequently used password is always a few clicks away. Even after I have sent a different password, I can quickly send the 'default' password with three long clicks followed by three short clicks. Just be sure your input focus is on the password field when you do this.
- A master password should be something that you can enter reasonably quickly using the rotary encoder, so if you're going to use a word think of one that's made up of characters from the beginning of the alphabet. For example; *cabbages* or *Abacus*. There are many other examples. You want a word or a combination of words and numbers that are not tedious to enter via the encoder. So I typically select a word that I can enter quickly followed by a four digit number. Of course you can enter anything you like, as long as it doesn't exceed 15 characters. Even a four digit pin might be secure enough for you.
- Remember to *Backup to EEprom* after changing attributes of existing credentials or after adding new credentials. I usually confirm that I can navigate through all existing accounts forwards and backwards before executing a *Backup to EEprom* operation, just to be sure the linked list that contains all of the credentials isn't corrupt. I haven't seen this problem for a long time.
- Before changing the master password make sure that you have a fresh backup in PasswordPump CSV format on hand (and hopefully encrypted). Immediately after performing a *Change Master Password* operation, confirm that you can navigate through all the accounts forwards and backwards, then perform a *Backup to EEprom* operation. If for some reason your credentials look corrupted after a *Change Master Password* operation (and before you backup), *Restore EEprom Backup* will restore your credentials to the primary EEprom with the original master password. You can also import the latest PasswordPump formatted CSV file via the PasswordPumpGUI after a *Factory Reset,* if necessary.

- Before removing the EEprom chip(s) from the device, power it off by unplugging it from your computer.
- Instead of executing a *Restore EEprom Backup* operation you can carefully swap the positions of the EEprom chips instead, and then *Backup EEprom*.
- Maintain a third EEprom backup and secure it in a safe place. You can purchase extra 25LC512-I/P DIP8 chips on AliExpress, Amazon or Ebay. If your data are corrupted and you cannot *Restore EEprm Backup,* you can insert this backup into the primary EEprom position (the top chip). Don't forget to perform a *Backup EEprom* operation twice; once to create a new offline backup and once to create a new online backup, for safety.
- Use the PasswordPumpGUI to create a PasswordPump formatted .csv file. Encrypt this file and/or store it on an encrypted thumb drive, and store the thumb drive in a safe or safe deposit box (perhaps alongside your EEprom backup). If your PasswordPump's data becomes corrupted you can perform a *Factory Reset* operation and then Import Password Pump file via the PasswordPumpGUI. If you are diligent about backing up your credentials in this manner (and in the manner described in the previous bullet) you can use the PasswordPump as your system of record for all of your credentials.
- EEprom chips can be moved to another PasswordPump device and will continue to function without modification. The master password moves with the EEprom chip. The hashed master password and salt are stored in the external EEprom chips.
- If the Old Password field is empty, it's automatically populated with the existing password when the Generate password button is clicked in the PasswordPumpGUI or directly via the device. If the field is *not* empty, clicking on the Generate password button will *not* move the existing password to the Old Password field, but it will overwrite the existing password with the generated password. If the Old Password field is populated and you want the existing password to move to the Old Password field after clicking the Generate button (which would be the typical use case), then blank out the Old Password and move input focus off of that field (so that the change is saved to the device) *before* clicking on Generate. The Old Password field is intended to protect the user from the situation whereby a password change is being made, the Generate button is selected to generate a new password, but the application or website for which you're changing the password does not accept the newly generated password for any reason and you have therefore lost the currently active password. By proper use of the Old Password field, in this situation, you have not lost the currently active password, it is in the Old Password field, and you can still use it to continue trying to reset the password. If you have feedback concerning the way this works, let me know.
- Use of the PasswordPumpGUI currently requires the installation of Python 3.8. At some point in the future an .exe will be created so that this requirement can be removed.
- By design it's possible to remove and replace the 25LC512 EEprom chips. For example, if you *Backup/Restore->Backup EEprom,* you can then remove the lower EEprom chip, which is the secondary/backup EEprom (the one closest to the RGB LED), and put it aside for safe keeping. You'd then want to use a third EEprom in its place, reinserting it into the device with the correct orientation, i.e. with the small dimple on the chip closest

to the screen. Be careful when removing and inserting these chips, the legs are delicate and easily bent. The best way to pull the chips out is with a chip puller. Make sure the device is powered off whenever you're inserting or removing one of these chips. Double check to be sure the orientation of the EEprom chips is correct *before* you power on the device; if the orientation is backwards the chips will heat up and something will fail dramatically. Finally, after replacing the secondary/backup EEprom with a new 25LC256, execute *Backup/Restore->Backup* one more time.

- You've forgotten your master password and you want to reset the device (hopefully because you're going to import your credentials from a PasswordPump backup file), but you can't because you obviously can't login to the device. Do you remember what your setting is for failed Login Attempts? The default is 10. Try to login to the PasswordPump that many times and the device will automatically factory reset! Set the new master password after the factory reset, and now you can import your PasswordPump backup file via the PasswordPumpGUI.



- While an 8 character password using only lowercase equals 26^8 combinations and will crack in less than 2 minutes via a brute force attack, a 10-character passphrase with uppercase, lowercase, numbers, and symbols is 94^10 combinations and will take approximately 600 years to crack, according to *Random-ize, "How Long Would It Take to Hack My Password", https://random-ize.com/how-long-to-hack-pass/.* My master password would take 589 years to crack (with a computer, not a rotary encoder, and without the retry maximum that resets the device!). My AD password would take 3,718,234,074,674,426,000 years to crack via brute force attack. I think that's 3.7 quintillion. The sun will burn out in 5 billion years. Naturally, none of this will do you any good if you re-use your passwords. And yes, the advent of quantum computing will change these numbers significantly.
- Source code is located here: https://github.com/seawarrior181/PasswordPump_II
- Send any issues and suggestions to dan-murphy@comcast.net.

 last revision date: 2020-07-02

Compiling the Source Code in the Arduino IDE

If you want to hack around with the PasswordPump source code, or just if you want to burn changes to the released code via the Arduino IDE (as opposed to the BOSSA program, covered in the next section), there are a few things to keep in mind. First, you'll need to install the libraries listed in the following table. If you're not sure how to install libraries, look for resources online to help you, I'm not going to cover that here.

Libraries

URL	Version	Library Description
https://github.com/LennartHennigs/Button2	1.0.0	Allows you to use callback functions to track single, double, triple and long click; for the button on the rotary encoder
https://github.com/arduino-libraries/Keyboard (only install if you're using a US keyboard, otherwise read the section 'Specifying the Correct Keyboard.cpp File, below)	1.0.2	This library allows an Arduino board with USB capabilities to act as a US Keyboard.
https://github.com/Necr0tizing/ArduinoKeyboardLayouts (do not install)	(none provided)	This library contains files that support non-US keyboard layouts.
https://github.com/rweather/arduinolibs https://rweather.github.io/arduinolibs/index.html	(none provided)	To perform cryptography operations on Arduino devices; for hashing the master password and encrypting credentials.
https://github.com/adafruit/Adafruit_SSD1306	1.3.0	This is a library for Monochrome OLEDs based on SSD1306 drivers
https://github.com/adafruit/Adafruit_SPIFlash	3.1.1	for FAT filesystems on SPI flash chips
https://github.com/thijse/Arduino-CmdMessenger	4.0.0	A messaging library for the Arduino and .NET/Mono platform

Specifying the Correct Keyboard.cpp File

If you're using a US keyboard you can skip the instructions in this section. Otherwise read on. By default the PasswordPump only works with US keyboards. You can, however, create a version that works with one of the following keyboards: Czech, Danish, Swedish, Norwegian, Finnish, French, German or Spanish. This is accomplished by installing the library located here:

https://github.com/seawarrior181/PasswordPump_II/tree/master/Libraries entitled Keyboard-master-multilingual *in place of* the standard Keyboard library. Then replace the Keyboard.cpp file with the version associated with the keyboard of your choice. For example, if I want to use a Spanish keyboard, I'd delete ..arduino\libraries\Keyboard-master-multilingual\src\Keyboard.cpp and rename ..\arduino\libraries\Keyboard-master-multilingual\src\Spanish-Keyboard.cpp to ..\arduino\libraries\Keyboard-master-multilingual\src\Keyboard.cpp. Then follow the remaining instructions for Compiling the Source Code in the Arduino IDE.

Fixing CmdMessenger

It's necessary to make an edit to the Arduino-CmdMessenger library so that it will compile. On line 492 of CmdMessenger.cpp, change

```
        return '\0';
 to
        char *str;
        *str = '\0';
        return str;
```

Fixing Adafruit_SSD1306 (optional)

Change Adafruit_SSD1306::begin in Adafruit_SSD1306.cpp to suppress display of the Adafruit splash screen on the SSD1306 display. Comment out lines 458 - 464 so that it looks like this:

```
  clearDisplay();
/* Remove the following 7 lines of code to suppress displaying the Adafruit splash screen
  if(HEIGHT > 32) {
    drawBitmap((WIDTH - splash1_width) / 2, (HEIGHT - splash1_height) / 2,
      splash1_data, splash1_width, splash1_height, 1);
  } else {
    drawBitmap((WIDTH - splash2_width) / 2, (HEIGHT - splash2_height) / 2,
      splash2_data, splash2_width, splash2_height, 1);
  }
*/
  vccstate = vcs;
```

Making the Correct Selections in the Tools Menu for the Adafruit ItsyBitsy M4

If you're not sure if you have an ItsyBitsy M4 or M0, under Tools select "Get Board Info". Under Tools do the following:

Set the Board: "Adafruit ItsyBitsy M4 (SAMD51)".
Cache: "Enabled"
CPU Speed: "120 MHz (standard)"
Optimize: "Small (-Os) (standard)"
Max QSPI: "50 MHz (standard)"
USB Stack: "Arduino"

Debug: "Off"
Port: "COM%% Adafruit ItsyBitsy M4 (SAMD51)"
Programmer: "Arduino as ISP"

(Where %% is the correct port number for your PasswordPump)

Making the Correct Selections in the Tools Menu for the Adafruit ItsyBitsy M0

If you're not sure if you have an ItsyBitsy M4 or M0, under Tools select "Get Board Info". Under Tools do the following:

Set the Board: "Adafruit ItsyBitsy M0".
USB Stack: "Arduino"
Debug: "Off"
Port: "COM%% Adafruit ItsyBitsy M0"
Programmer: "Arduino as ISP"

(Where %% is the correct port number for your PasswordPump)

Selecting the Correct Pre-compiler Directives

Note that in PasswordPump_v_2_0 there are several precompiler directives starting where the main block of comments ends, around line 789. The are as follows:

```
//#define __LEFTY__   // Turn this on if you have a "lefty" rotary encoder
//#define __SAMD51__  // Turn this on for Adafruit ItsyBitsy M4
//#define __SAMD21__  // Turn this on for Adafruit ItsyBitsy M0
```

- Uncomment __LEFTY__ if the rotary encoder is proceeding backwards through the alphabet when it is turned clockwise, instead of forward through the alphabet (the desired behavior).
- Uncomment __SAMD51__ if your board is an Adafruit ItsyBitsy M4.
- Uncomment __SAMD21__ if your board is an Adafruit ItsyBitsy M0.

Compiling and Uploading the Program

Double click on the reset button on the bottom of the PasswordPump, and notice that the RGB LED fades blue bright and dim slowly. Once you do this the port number will change, so go back to Tools-->Port and select the correct port. Then select Sketch-->Upload.

Uploading the Latest Firmware to the PasswordPump via BOSSA

The PasswordPump arrives with firmware installed. If you want to upgrade the firmware because a newer version is released, there are three ways to do that. You can use the Arduino IDE in combination with the PasswordPump C++ source code and associated libraries (documented in the source code of the program as well as in the previous section), you can use the BOSSA tool from the command line, or you can use the BOSSA user interface, probably the easiest route. Note that you can burn new firmware without erasing the existing credentials stored on the EEprom chips. Burning the firmware using the Arduino IDE was covered in the previous section. Burning the firmware via BOSSA is covered below.

From BOSSA

You can download BOSSA from here: https://github.com/seawarrior181/PasswordPump_II or here https://github.com/shumatech/BOSSA/releases and install it on your MS Windows or Apple Mac OS X computer in the usual fashion. Then you can burn the firmware either from the **BOSSA** user interface (easiest) or via the **bossac** command line. First download PasswordPump_v_2_0.ino.bin from https://github.com/seawarrior181/PasswordPump_II/tree/master/v2_0_3 and save it to C:\temp.

Burning Firmware From the Command Line

Find where the **bossac** utility is installed on your system and modify the path below as per its location. Identify the port to which the ItsyBitsy/PasswordPump is connected (you can use the Device Manager or the Arduino IDE to do this) and substitute the correct port number in the command below. Double click on the reset button at the bottom of the PasswordPump. The RGB LED will slowly dim and brighten in blue over and over again. From a command window (<Alt><Esc>cmd<Return> in Windows) execute a command similar to the following:

```
C:\Users\username\AppData\Local\Arduino15\packages\arduino\tools\bossac\1.8.0-48-gb176e
ee\bossac -i -d --port=COM52 -U -i --offset=0x4000 -w -v
C:\Temp\PasswordPump_v_2_0.ino.bin -R
```

Make sure that the port and the offset are correctly specified. **The offset is always 0x4000**. You should observe output that reflects that the PasswordPump's firmware has been updated. Click the reset button on the device once to start using the PasswordPump.

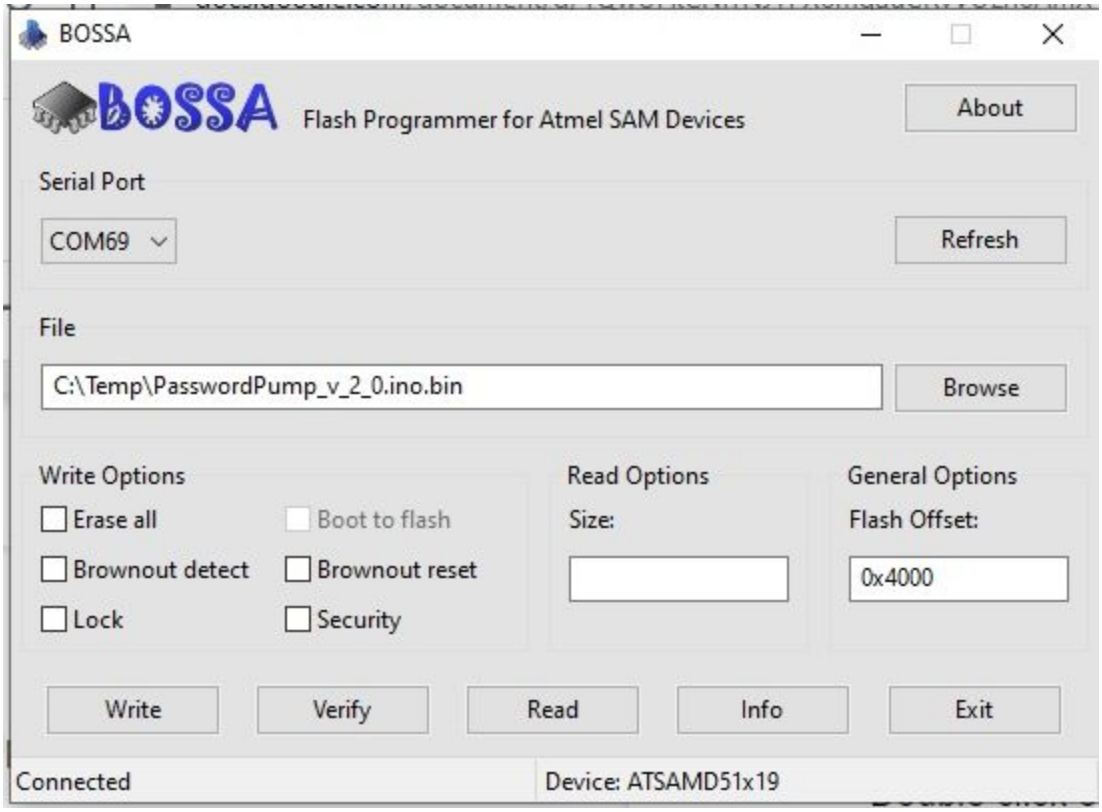
At this time it's also important to download the latest version of the PasswordPumpGUI.

Burning Firmware From the BOSSA GUI

Double click on the reset button on the PasswordPump so that the blue LED slowly dims and brightens in blue before burning the firmware. After starting up the **BOSSA** user interface, to burn the firmware, use all of the defaults except **specify a flash offset of 0x4000**. The file location should be C:\Temp\PasswordPump_v_2_0.ino.bin. Select the correct port. Click *Write*

to write the firmware to the device, then click *Verify* to verify that it was written correctly. Finally click the reset button on the PasswordPump once to start using it.

At this time it's also important to download the [latest version of the PasswordPumpGUI].



RGB Colors and Meanings

Color	Meaning
Green	Logged in/Logged in
Orange	Backing up EEprom memory
Alternating Blue and Red	Initializing EEprom or auto logout pending
Purple	Editing with computer via PasswordPumpGUI, Sending creds, backing up to EEprom.
Red	Error backing up or initializing EEprom, failed login attempt(s).

Yellow	Error backing up or initializing EEprom
Yellow fast flash	Changing master password
Cycling through all colors	Not logged in

Error Codes

These error codes are observed on the PasswordPump device, typically on the third line, when something goes wrong. The screen will invert (to get your attention) and the error code will display on the third line for two seconds. If you see any of these codes you should report the incident to dan-murphy@comcast.net, providing as much detail as possible.:

000 - SSD1306 allocation failed (only visible via serial)
001 - Error navigating Off On menu
002 - Error navigating main menu
003 - Error navigating edit credentials menu
004 - Error navigating send credentials menu
005 - Error navigating settings menu
006 - Error showing credential values
007 - Unrecognized event
008 - Invalid state when showing Off On menu
009 - Invalid login attempt maximum
010 - Out of space
011 - Corruption found
012 - Out of space during import
013 - Failed to open file for import
014 - Failed to mount FAT file system during import
015 - Failed to initialize flash during import
016 - Invalid RGB LED Intensity position
017 - Invalid maximum login attempt count
018 - Invalid logout timeout value
019 - Invalid keyboard, show password or decoy password value
020 - Account name keeps encrypting to 255 in first char during import
021 - User name is too long on import
022 - Password is too long on import
023 - Web site is too long on import
024 - Account name is too long on import
025 - Invalid group specified
026 - Invalid search group specified
027 - Invalid group menu item specified
028 - Invalid state during event single click
029 - Invalid state encountered during rotate counter clockwise event

030 - Invalid state encountered during rotate clockwise event
031 - Empty credentials found in linked list
032 - Corrupt linked list
033 - Corrupt linked list in FindAccountPos.
034 - Failed to initialize flash during PasswordPump CSV file import
035 - Group length is greater than one
036 - Too many fields found in PasswordPump CSV file during import
037 - Failed to open PasswordPump CSV file for import
038 - Invalid position in file menu
039 - Encrypted account name starts with 255, fixing...
040 - Invalid position when returning to a find by group menu
041 - Corrupt link list encountered while counting accounts
042 - Invalid position when returning to settings menu
043 - Invalid group number when customizing groups
044 - Invalid category number when customizing groups from PasswordPumpGUI

Datasheets

If you're interested in hacking around with the PasswordPump the locations for some of the datasheets might be helpful:

AdaFruit ItsyBitsy M4 (32-bit ARM®, SAMD51 Cortex®-M4F MCU)
Data Sheet: *http://ww1.microchip.com/downloads/en/DeviceDoc/60001507E.pdf*
https://learn.adafruit.com/introducing-adafruit-itsybitsy-m4/downloads
MICROCHIP - 25LC512-I/P - 512K SPI™ Bus Serial EEPROM DIP8, one primary one backup.
Data Sheet: *http://ww1.microchip.com/downloads/en/DeviceDoc/20005715A.pdf*
SSD1306 I2C LED display 128x32 pixels.
Data Sheet: *https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf*
https://www.vishay.com/docs/37894/oled128o032dlpp3n00000.pdf

Why PasswordPump?

Why should I use the PasswordPump instead of a more traditional password manager? There are several interesting discussions in the links provided below. For me it boils down to speed and security. With practice I'm able to quickly locate the credentials for any account to which I need to login, and I have many of them. I feel more secure knowing that my credentials are

encrypted and stored in only one place; on a device I can hold in my hand and store in my safe. Someone who wants access to my credentials needs to take possession of the device and needs to crack the encryption. That requires a very high level of motivation.

Why you shouldn't store passwords in your browser

Most web browsers offer to store your passwords for you. This might seem like an ideal way to keep track of your passwords – but it can actually be a bad idea. Here are some reasons why:

- The password security on browsers isn't that great – even if you are using a secure browser. Sometimes, these passwords are stored in plaintext. There are also tools available online that can give hackers access to your computer (either physically or remote access schemes) and view/steal passwords stored in the browser.
- Your browser will only record the username and password you enter into a web page. It won't help you generate a password, or tell you if the password is strong, or remind you that you already used this same password on 10 other pages.

From https://www.techspot.com/news/83704-best-password-managers.html:

How safe are password managers? Good discussion:
https://security.stackexchange.com/questions/45170/how-safe-are-password-managers-like-lastpass
More password discussion:
https://siliconangle.com/2020/01/20/lastpass-suffers-outage-first-denied-quietly-confessed/
https://www.helpnetsecurity.com/2020/03/09/passwords-data-breaches/
https://fossbytes.com/1-2-million-microsoft-accounts-hacked-made-same-mistake/
https://www.forbes.com/sites/zakdoffman/2020/03/07/microsoft-confirms-really-really-high-hacking-threat-for-millions-of-users-heres-what-you-do-now/#22d5867b9b66
https://www.foxnews.com/tech/5-ways-improve-passwords-not-get-hacked
https://techxplore.com/news/2020-03-expose-vulnerabilities-password.html

---

Known Defects

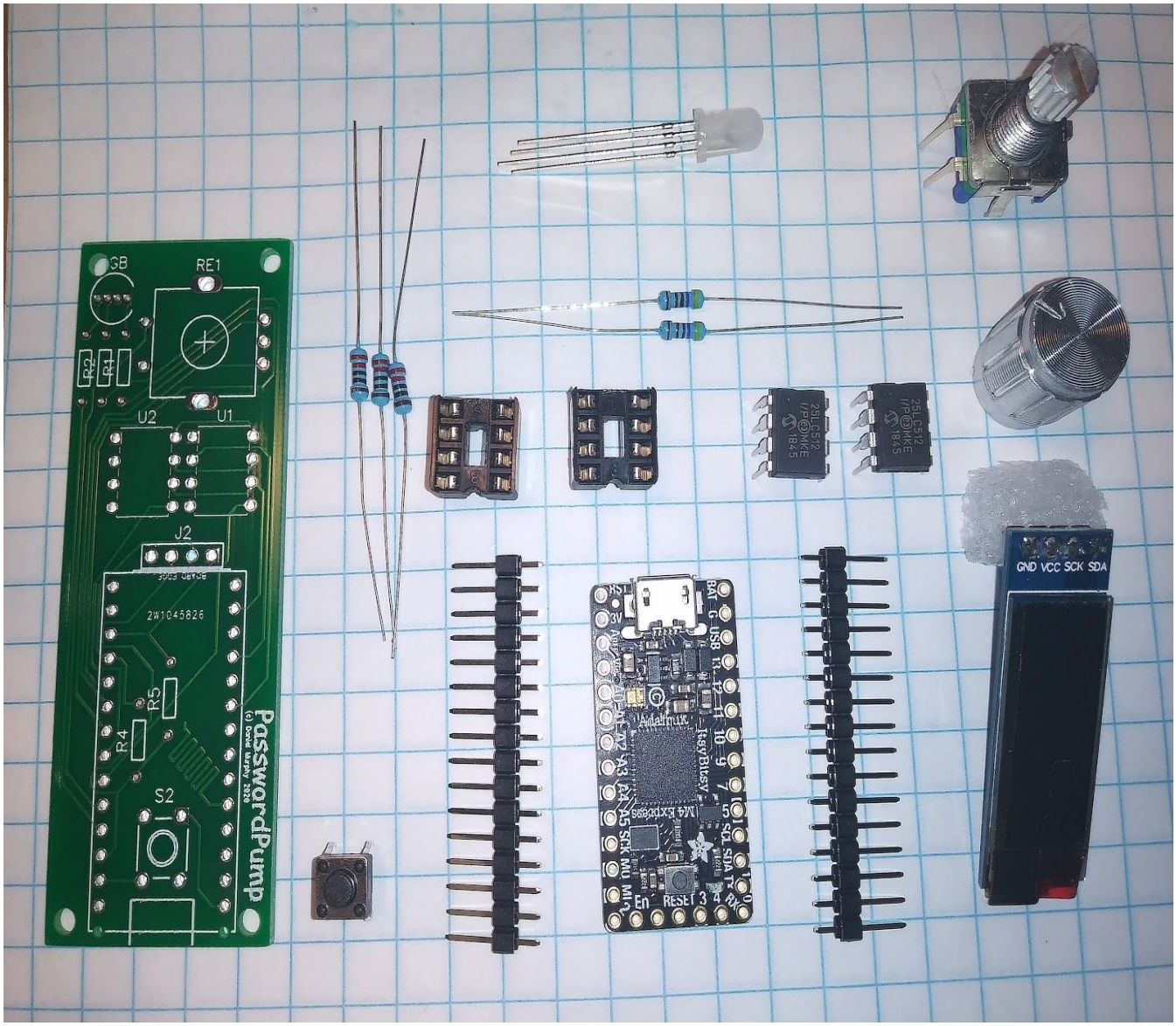
1. In the PasswordPumpGUI, if an account name contains a comma, and you visit the field, after exiting the PasswordPumpGUI and reloading all of the accounts, the comma has changed into a hashtag and all of the remaining fields are blank.
2. Embedded quotes in a CSV import file are not getting saved to the field.
3. When you import credentials with <CR><LF> in the account name bad things happen.
4. When entering an account name 29 chars long via keyboard, nothing gets entered.
5. Fixed in 2.0.4: entering the decoy password does not factory reset the device.
6. Found 5/10/2020, fixed in 2.0.4: Via PasswordPumpGUI Insert, then <Alt><Tab> to another application. Upon returning to PasswordPumpGUI the Account Name is

"Unknown". Set focus to another account, the PasswordPump and PasswordPumpGUI freeze. Close the PasswordPumpGUI window and long click on the PasswordPump. Now there is only one account in the PasswordPump. Restore from secondary EEprom.

Assembly Instructions

1) Kindly read through all of these instructions and gather all of the parts below before starting assembly:

1 Custom PCB
3 220 ohm resistors
2 4k7 ohm resistors
1 button
2 8 leg IC DIP sockets
1 5mm RGB LED, diffused
2 25LC512 EEproms
1 15mm rotary encoder
1 SSD1306 I2C OLED Display Module 0.91 Inch
1 ItsyBitsy M4
2 sets of mail headers, 14 pins per header
1 USB cable

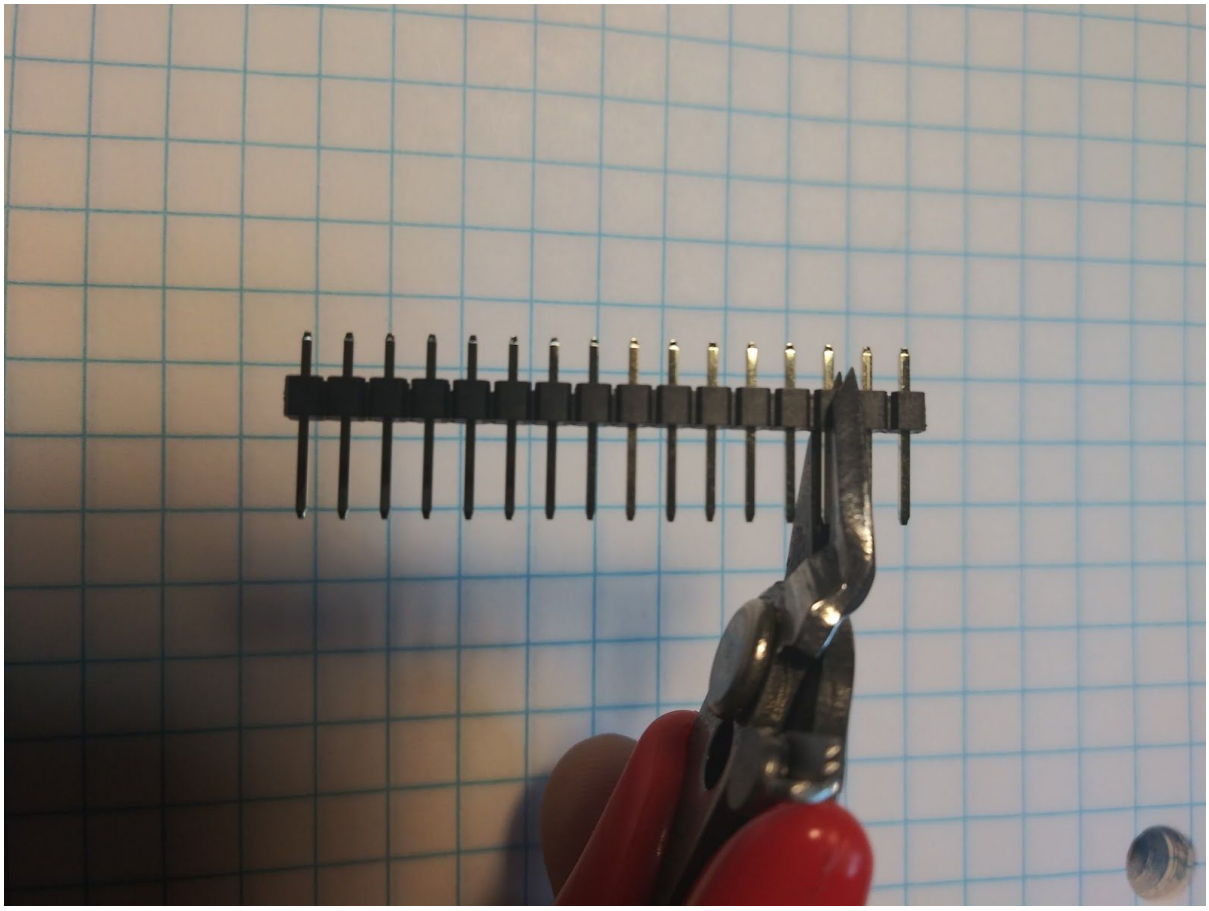


All of the parts supplied in the kit (USB cable excluded from picture).

2) Tools Needed:

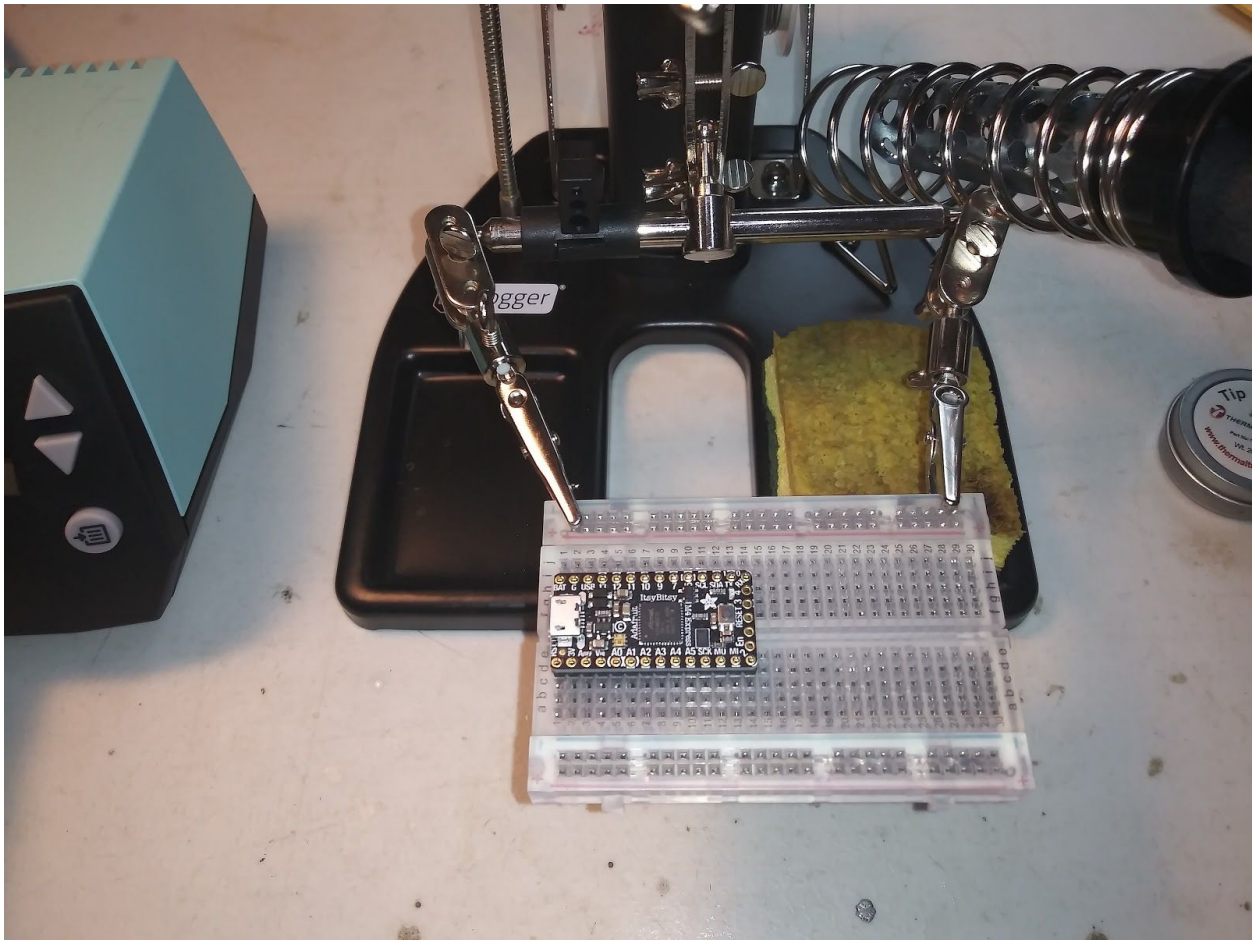
- Soldering Iron (set to ~720 degrees Fahrenheit)
- Wire cutters
- Desoldering tool (might be necessary for removing solder bridges)
- Helping hands (for soldering parts)
- Multimeter (for checking for absence of continuity on RGB LED leads)
- Jeweler's loupe (for visually inspecting solder on the RGB LED and elsewhere)

3) Snip each set of male headers for the ItsyBitsy with your wire cutters so that they are 14 pins long.



Snipping the excess off of the male headers.

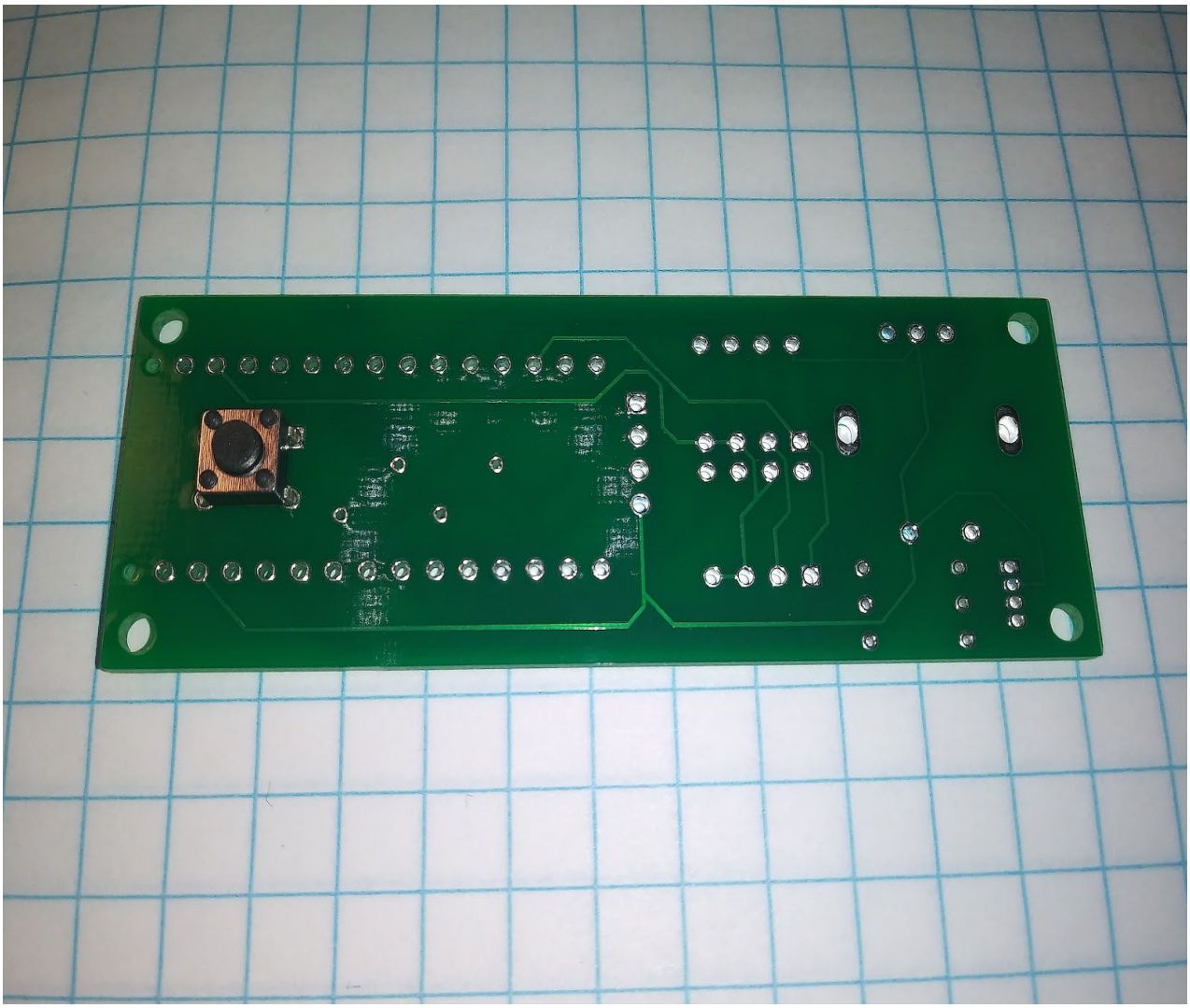
4) Place the male headers on the ItsyBitsy and then assemble the headers and the ItsyBitsy onto a small breadboard so that when you solder the headers to the ItsyBitsy they are properly aligned; i.e. they will fit into the custom PCB.



Using a breadboard to solder the male headers onto the ItsyBitsy.

5) Solder the male headers onto the ItsyBitsy (not the PCB!). Be careful not to overheat the pins on the ItsyBitsy. It's a good idea to alternate sides and alternate pins so that one section of the board doesn't get too hot. I set my iron to 720 degrees Fahrenheit. Do not attach the ItsyBitsy to the custom PCB yet.

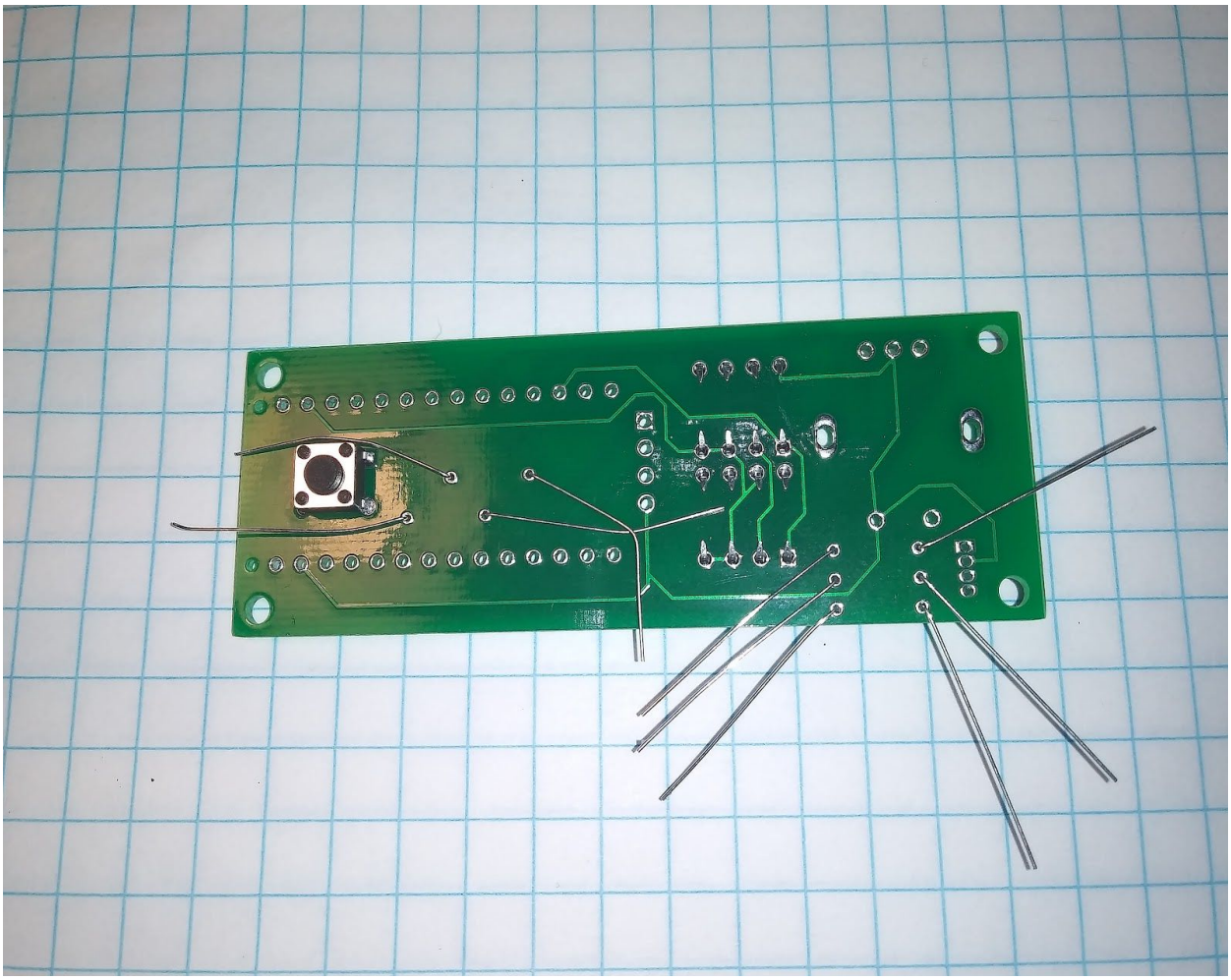
6) Attach the button to the custom PCB. Attach it to the BOTTOM of the PCB (i.e. the side opposite the side that's marked PasswordPump). If you do not put the button on the correct side it will be impossible to solder the ItsyBitsy to the custom PCB.



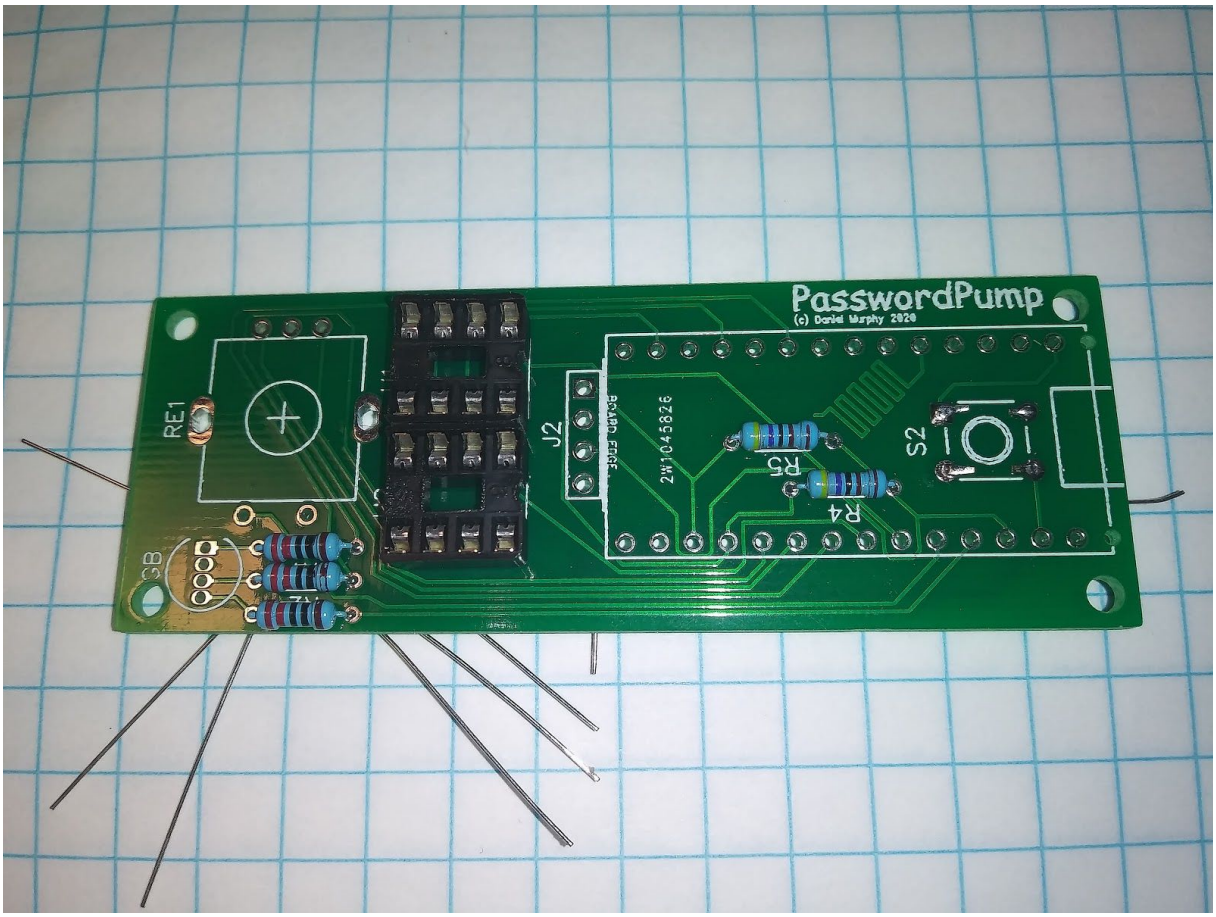
Solder the button to the bottom of the PCB!

7) Solder the button in place and then turn over the PCB.

8) Attach the resistors and the 8 leg IC DIP sockets to the custom PCB. As you attach each part to the PCB turn the PCB over and bend the legs so that the part remains affixed to the PCB. When placing the resistors, they should be flush with the board, not elevated in any way. R1, R2, and R3 (R3 is unlabeled but is above R1) are the 220ohm resistors (red, red, black, black, brown), R4 and R5 are the 4k7ohm resistors (these are the I^2C pullup resistors, yellow, purple, black, brown, brown). When placing the 8 leg IC DIP sockets (U1 & U2), orient the dimple so that it faces the ItsyBitsy. This isn't critical but it's the right way to do it.

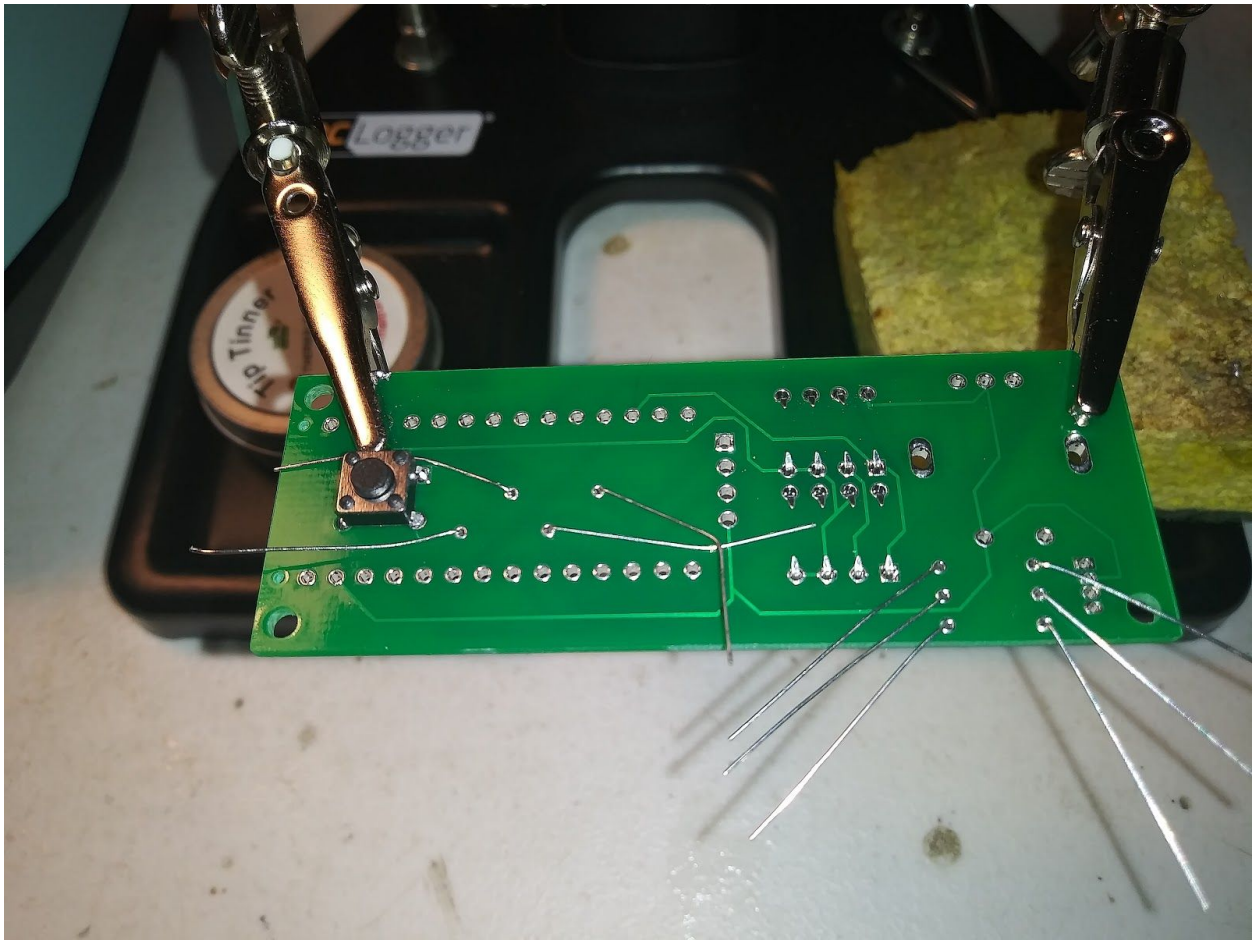


Looking at the bottom of the PCB after resistors and DIP sockets have been placed.



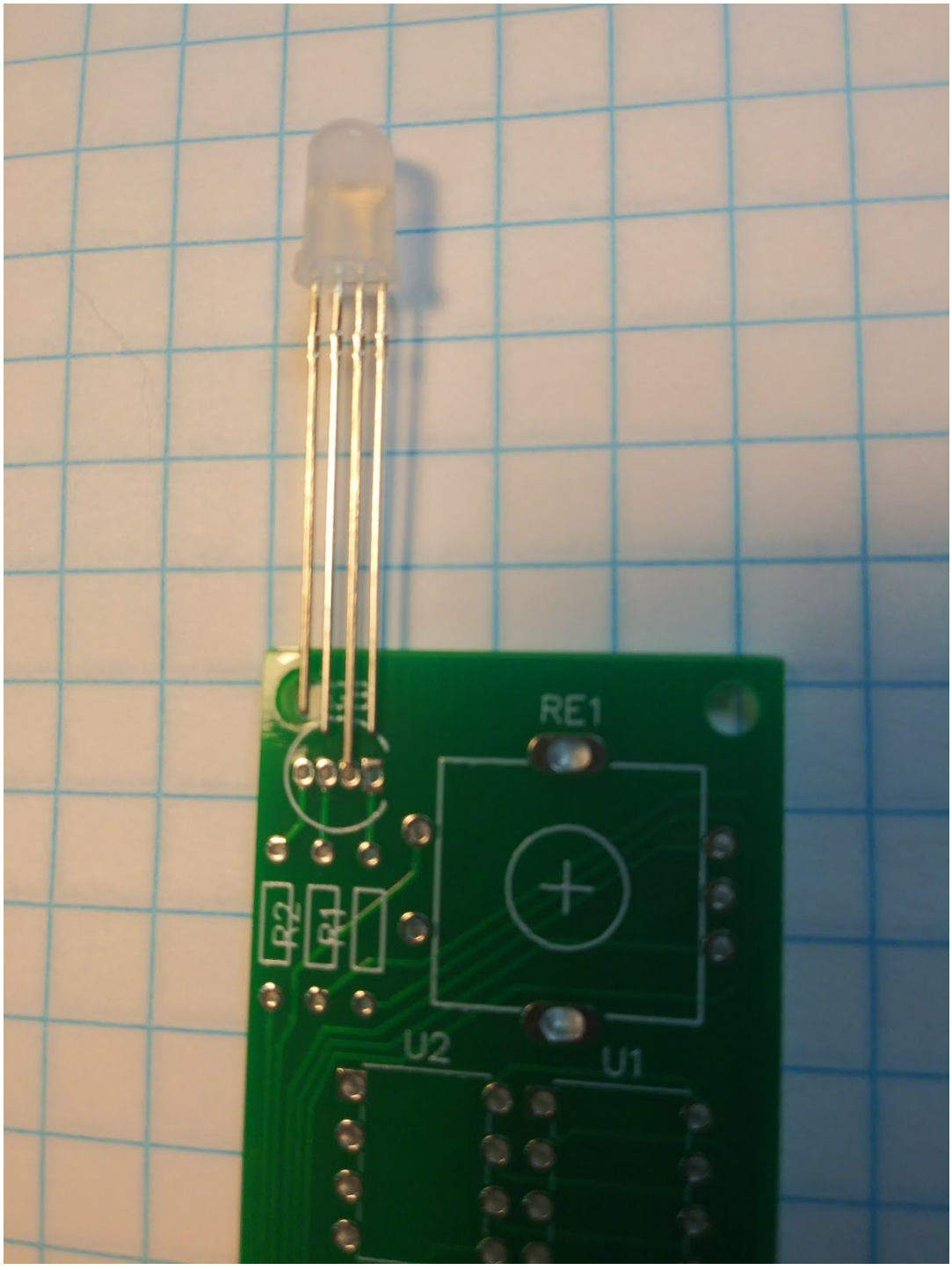
The top of the device after resistors and DIP sockets have been placed.

9) Solder the attached components in place, and snip the excess resistor legs.



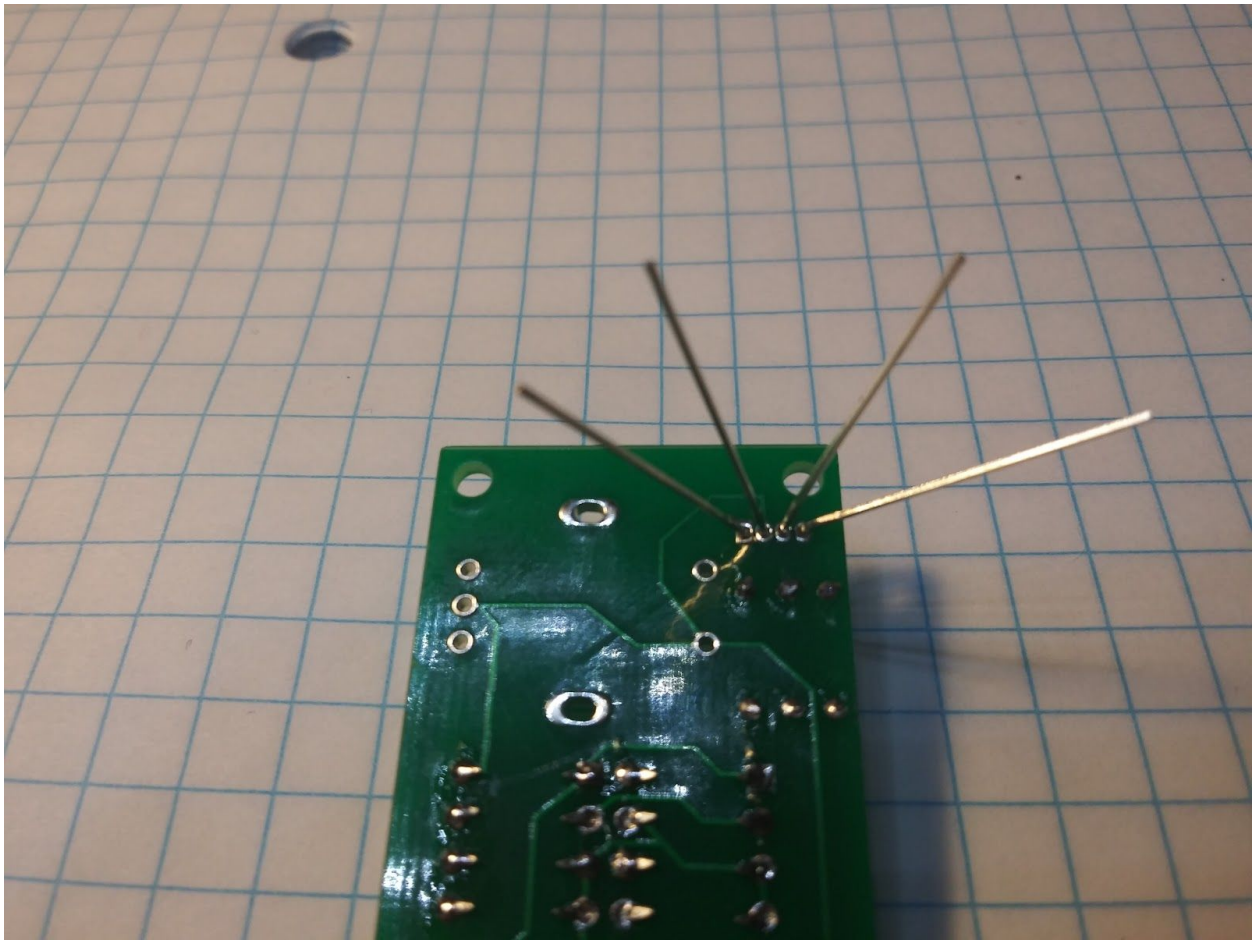
Preparing to solder the parts in place.

10) Attach the RGB LED to the custom PCB. Be careful to orient the RGB LED correctly; orient the leads so that the longest lead goes through the hole that's third from the left and second from the right. That's the same hole that does not have a visible trace on the top of the PCB, it's on the bottom instead, and it's the negative lead. See the picture below.



Insert the RGB LED into the PCB with the correct orientation.

11) Insert the RGB LED all the way in and spread the leads on the opposite side so that the LED doesn't fall out.



Spreading the leads of the RGB LED so it won't fall out and can be soldered.

12) Solder the RGB LED into place. This is the most difficult thing to solder in the kit because the leads are so close together and it's very easy to bridge the solder between them. I work from the inside out; first soldering the 2nd lead by placing the iron between the 1st and 2nd leads. I use a minimum amount of solder. Then I solder the 3rd lead by placing my iron between the 3rd and 4th leads. Then I solder the 1st lead by placing my iron on the outside of it (so that it is not in contact with the 2nd lead). Finally I solder the 4th lead by placing my iron on the outside of it (so that it is not in contact with the 3rd lead). Then, before snipping the leads, I use my multimeter to check for continuity between leads 1 and 2, 2 and 3, and 3 and 4. If there's no continuity you're in good shape. If you find that there is continuity then there's a solder bridge between the respective leads. Use patience, your soldering iron and optionally, the desoldering tool to remove the bridge(s). Re-check for continuity and repeat if necessary. Snip the excess leads. It also helps to examine the solder joints through a jeweler's loupe or similar magnification device.

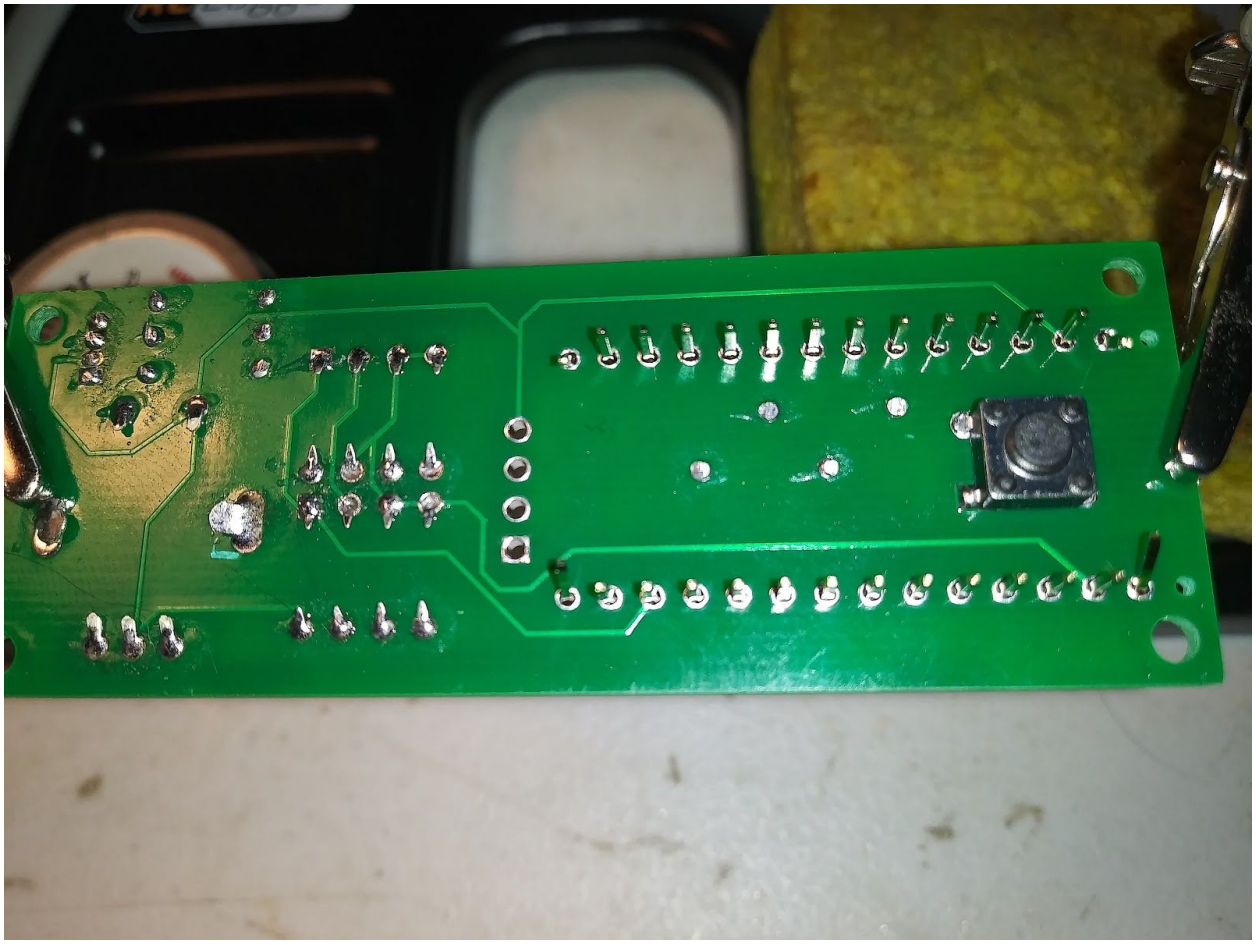
Checking the continuity on the RGB LED leads.

13) This can be a difficult step. Align the legs on the rotary encoder with the holes in the PCB (RE1). You might need to gently bend the legs on the rotary encoder to get them to line up with the holes. Gently insert the rotary encoder into place, and bend the leads over on the back so that the encoder stays in place. Be careful not to bend the legs too many times or they will break off. Be patient it can take some trial and error to get everything to line up correctly.

14) Now turn over the PCB and solder the rotary encoder in place.

15) Solder the ItsyBitsy in place. Be sure to orient it correctly, the micro USB port should be at the end of the PCB as depicted on the PCB's silk screen. After inserting the ItsyBitsy through the breadboard, slightly bend the leads at the four corners so that the ItsyBitsy stays in place when you turn the PCB over to solder it. There is no need to snip the leads on the ItsyBitsy, and, in fact, leaving them there might help you to refrain from accidentally pressing the reset button on the bottom of the PCB. When soldering the ItsyBitsy in place be careful not to melt the button with the soldering iron. Solder every other pin so that one section of the board

doesn't get too hot; then come back and solder the skipped pins. Closely inspect your work with the jeweler's loupe to be sure there are no solder bridges between the pins.

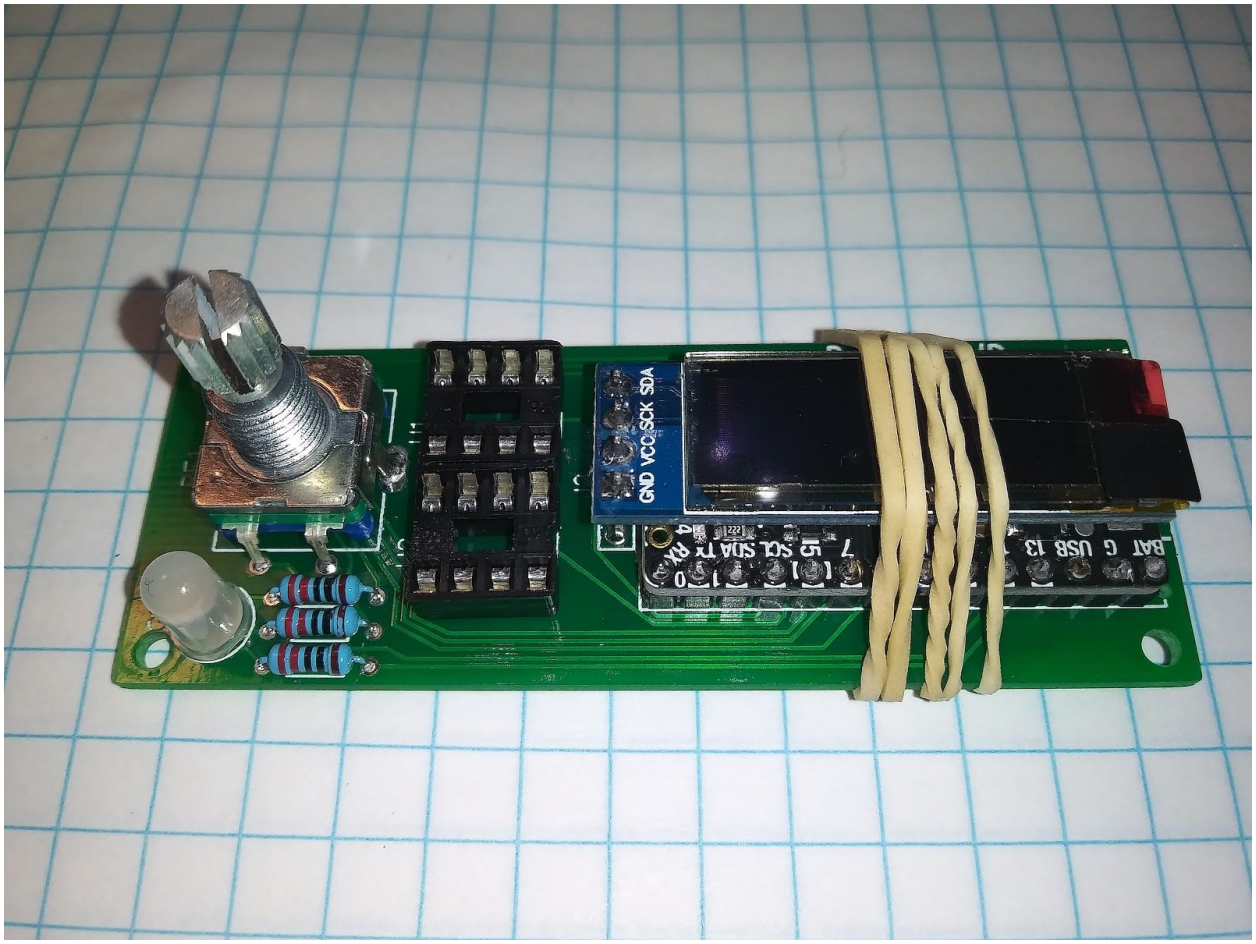


Note the four corner legs of the ItsyBitsy are bent so that it stays in place for soldering.

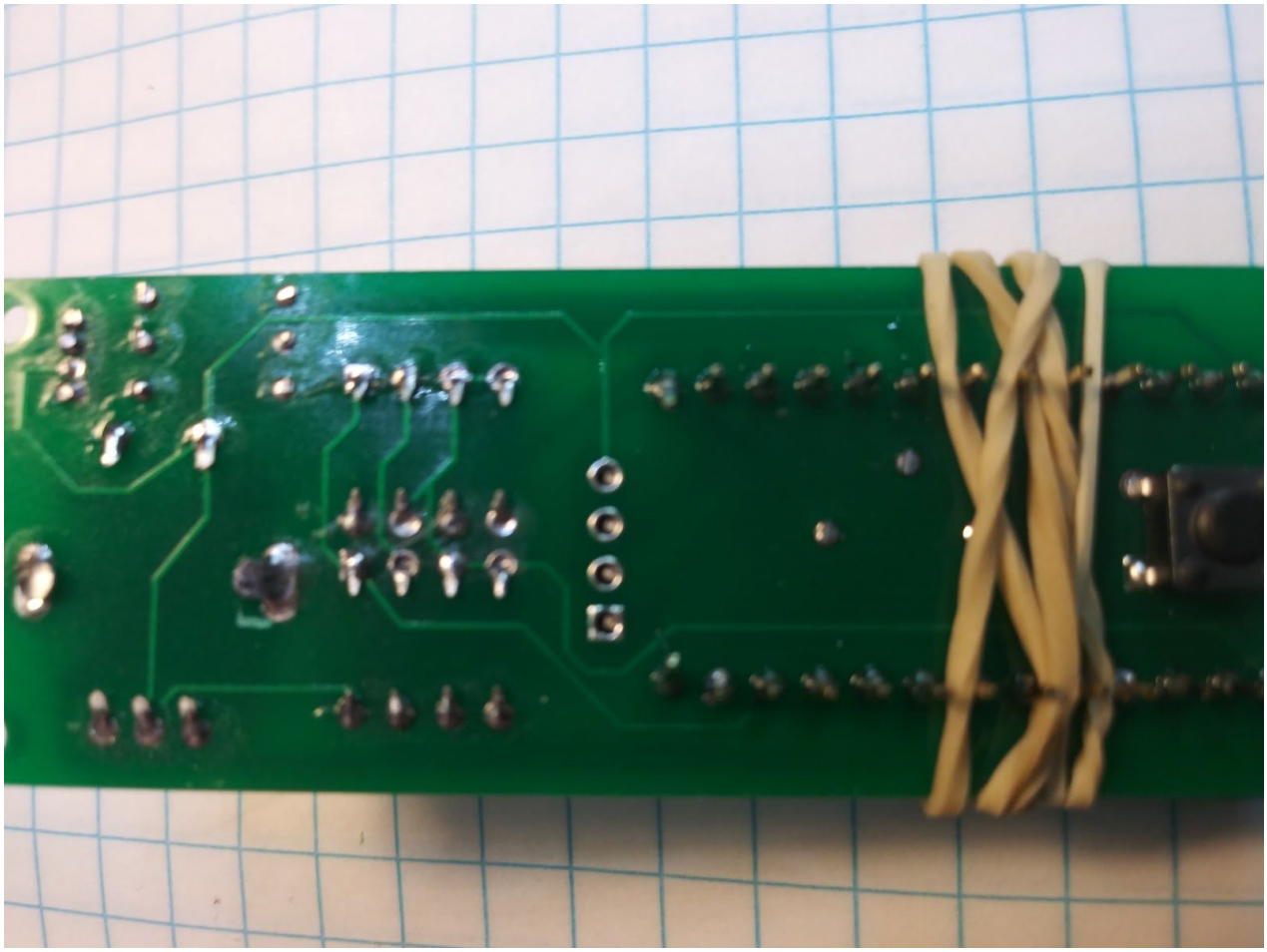
16) If your kit arrived without the headers soldered to the SSD1306 display, solder the headers onto the display now. Be careful to keep the headers square (at a 90 degree angle to the display itself). Using a breadboard and a quarter to keep things square can be helpful.

Using a breadboard and a quarter to solder headers onto the SSD1306 display.

17) Solder the SSD1306 display in place (J2). Attach it to the ItsyBitsy using an elastic. The leads from the display should barely protrude from the bottom of the PCB, if at all. Be certain that the display is not on so tight that it permanently presses the reset button on the ItsyBitsy.

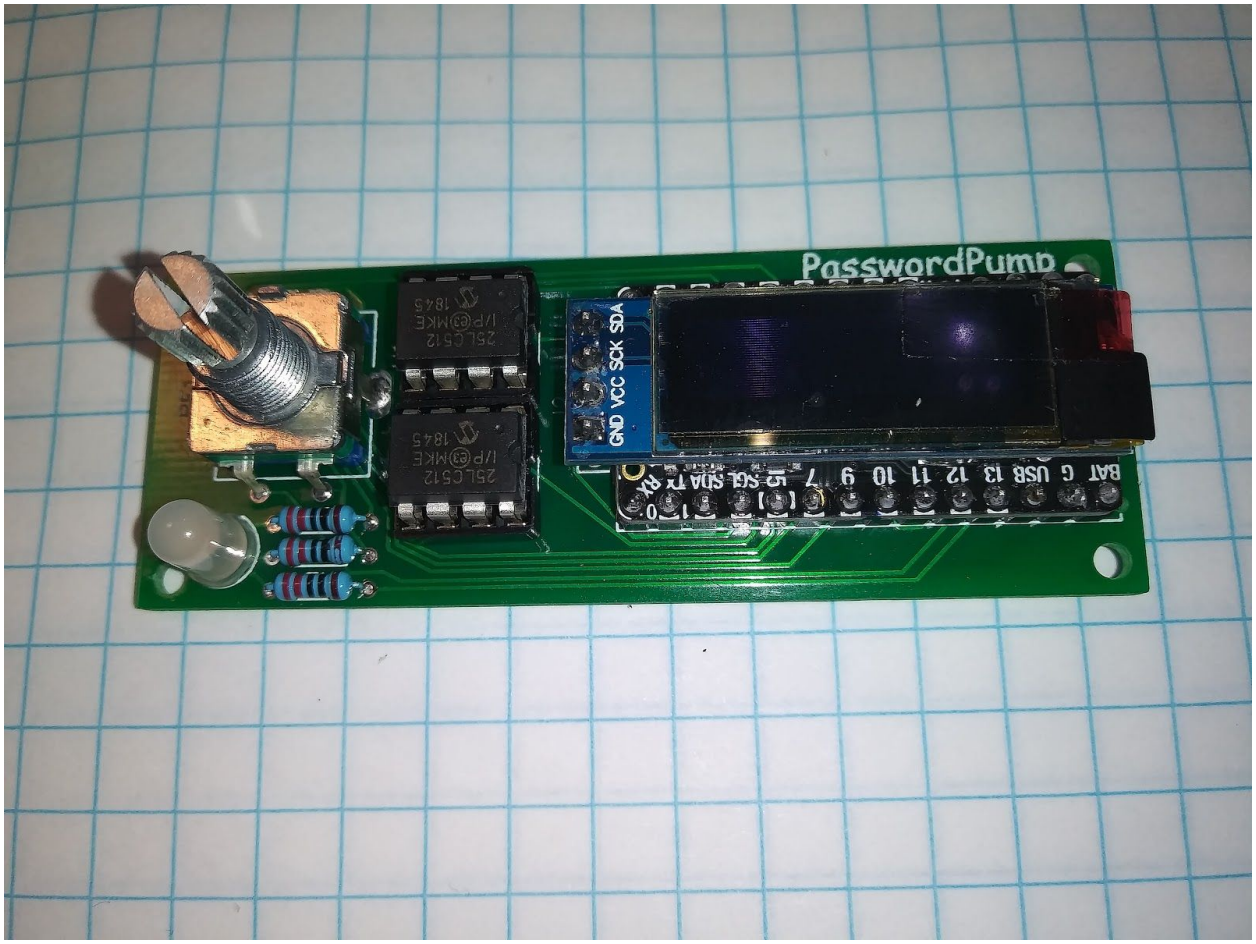


Fixing the display in place with an elastic for soldering.



Solder the display in place where the male headers are barely poking through the PCB.

18) Insert the 25LC512 EEprom chips into the IC DIP sockets. Orient the chips correctly by checking to be certain the dimples on the chips are closest to the display. You'll need to bend the legs of the chips inward slightly so that they will insert into the IC DIP sockets.



Mostly assembled PasswordPump without the knob. Note the orientation of the 25LC512 EEprom chips.

19) Insert the plastic knob onto the rotary encoder by pushing it down onto the encoder. If you ever remove the knob from the encoder remember to hold the encoder, not the PCB, when pulling the knob off. Otherwise you will rip the encoder off of the board. Ensure that you can actuate the encoder with the button on it. If for any reason the button press on the encoder doesn't feel right (like it's not clicking), carefully remove it, use the device without it, and let me know. Depending on how tight the fit is, you can also refrain from pushing the knob all the way onto the encoder so that it clicks correctly when depressed. The rotary action on the encoder is a little stiff at first, but it loosens over time.

20) Firmware is already burned onto the ItsyBitsy, so you should be able to plug in the PasswordPump into your computer's USB port via a USB to micro USB cable and start working with it. Goto **Initial Setup** on page 2. If there are any issues with the device or the documentation please inform me.

"Lefty" Rotary Encoders

You might notice, after you've built your PasswordPump and burned the firmware yourself, that the rotary encoder isn't behaving as expected, that is, when you rotate it clockwise it proceeds backwards through the alphabet and through the numbers instead of forwards through the alphabet. The PasswordPump functions perfectly fine like this but you may wish to straighten it out. Unfortunately to do so you need to modify the source code, recompile it, and re-upload it to the PasswordPump. To modify the source code, proceed to about line 769, at the very beginning of the "Includes/Defines" section, and uncomment the line with the __LEFTY__ pre-compiler directive. E.g.

Change

```
//#define __LEFTY__
```

to

```
#define __LEFTY__
```

Compile and upload the code (as described here) and you should be good to go.

Contact Information

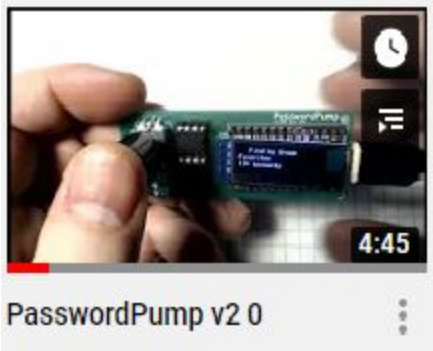
Dan Murphy
dan-murphy@comcast.net
5volts.org

Purchasing PasswordPump

Visit the PasswordPump on Tindie.com to purchase another PasswordPump.
https://www.tindie.com/products/19375/

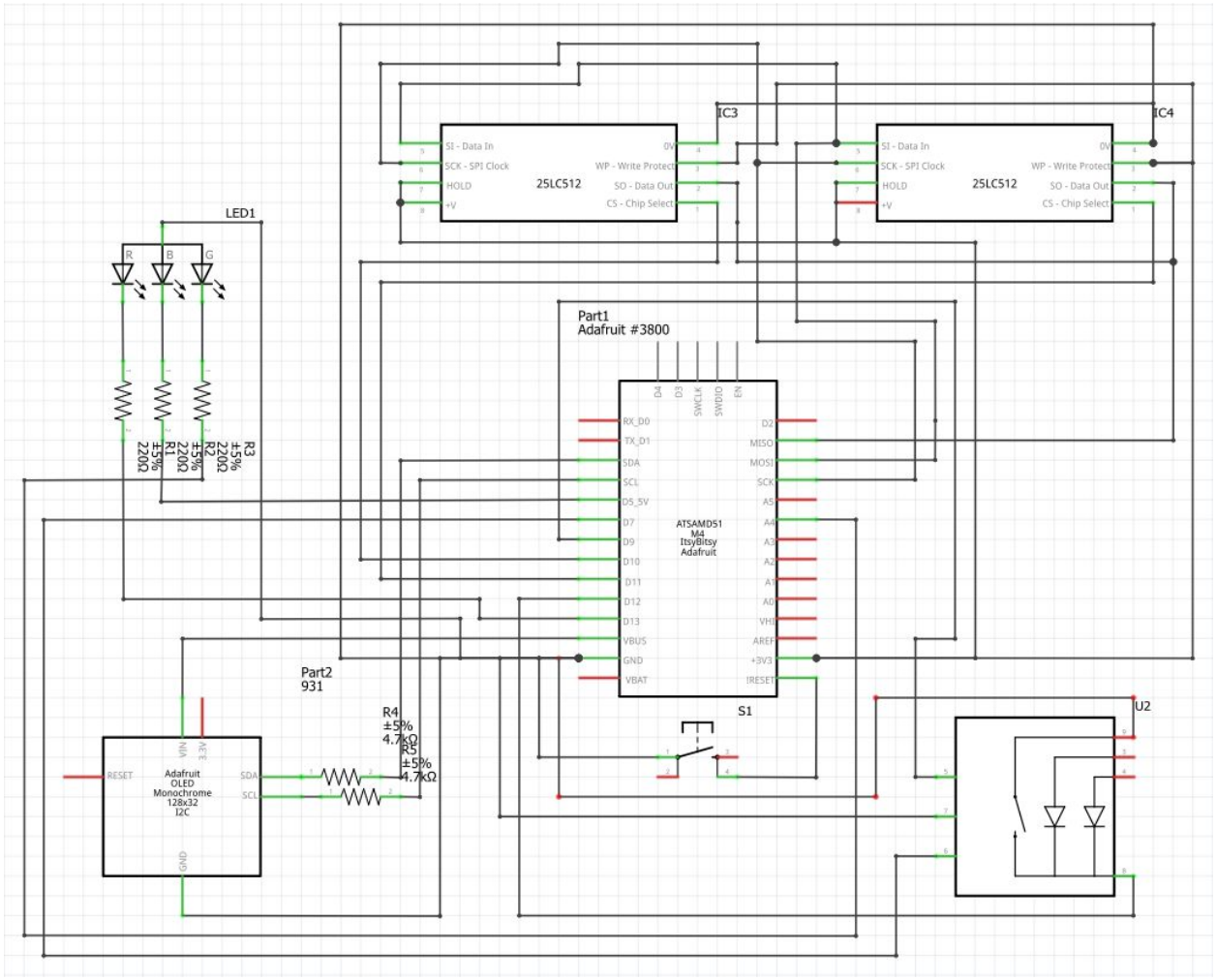
 last revision date: 2020-07-02

Video

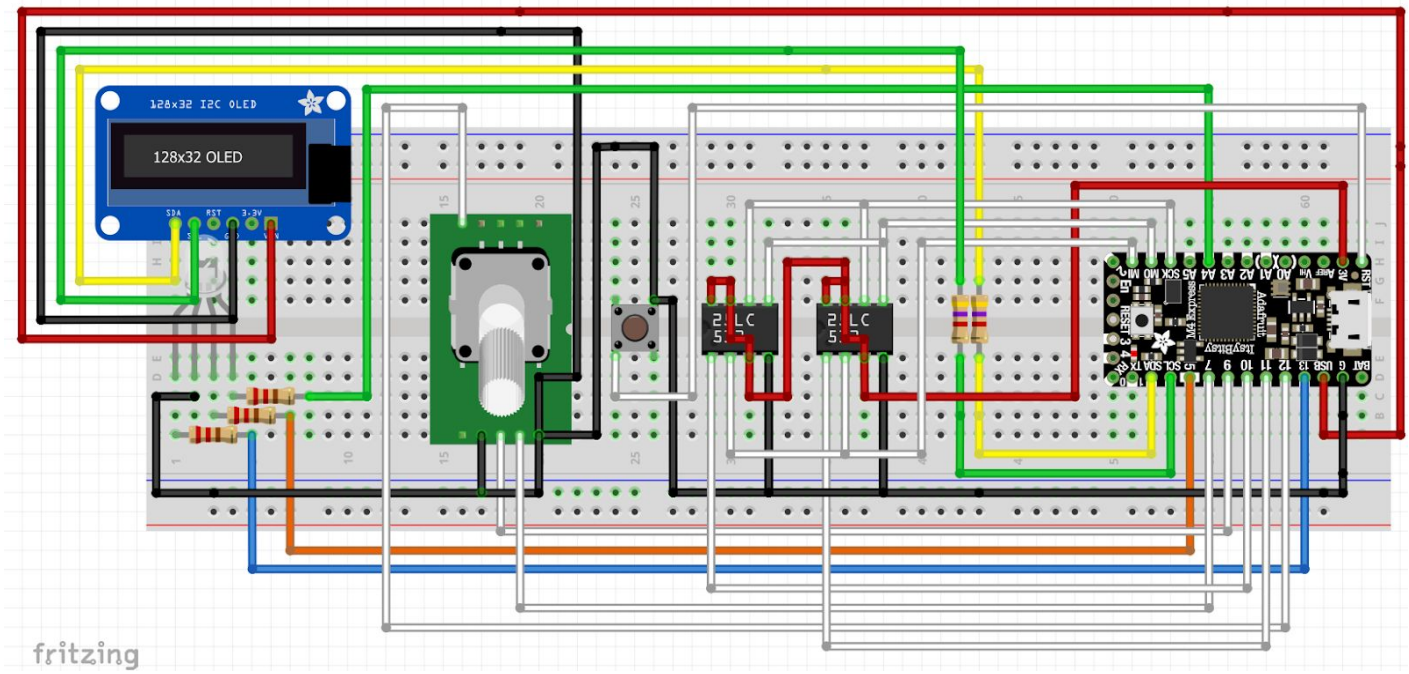


Click here (https://youtu.be/f4Iukt5VDUo) to see the PasswordPump v2.0 on YouTube.

Schematic



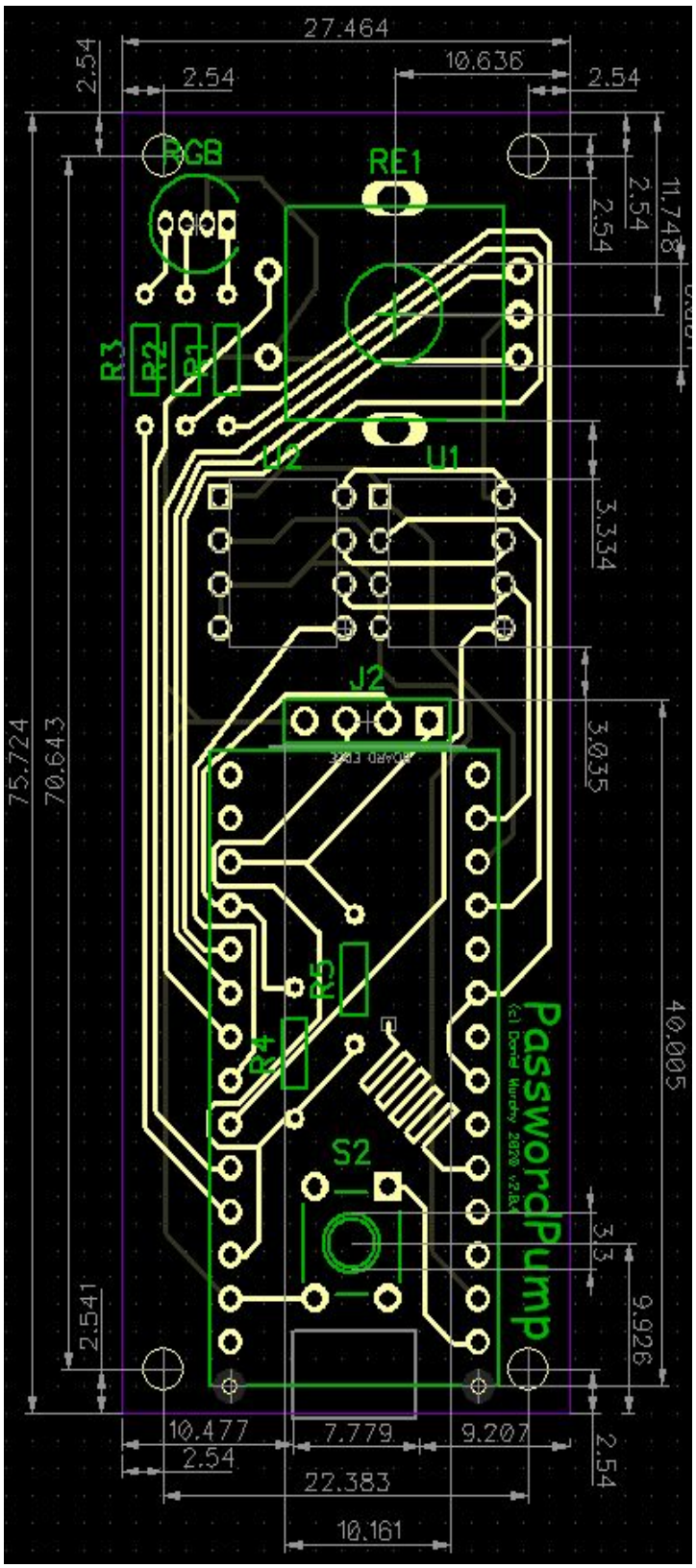
Fritzing Breadboard Layout



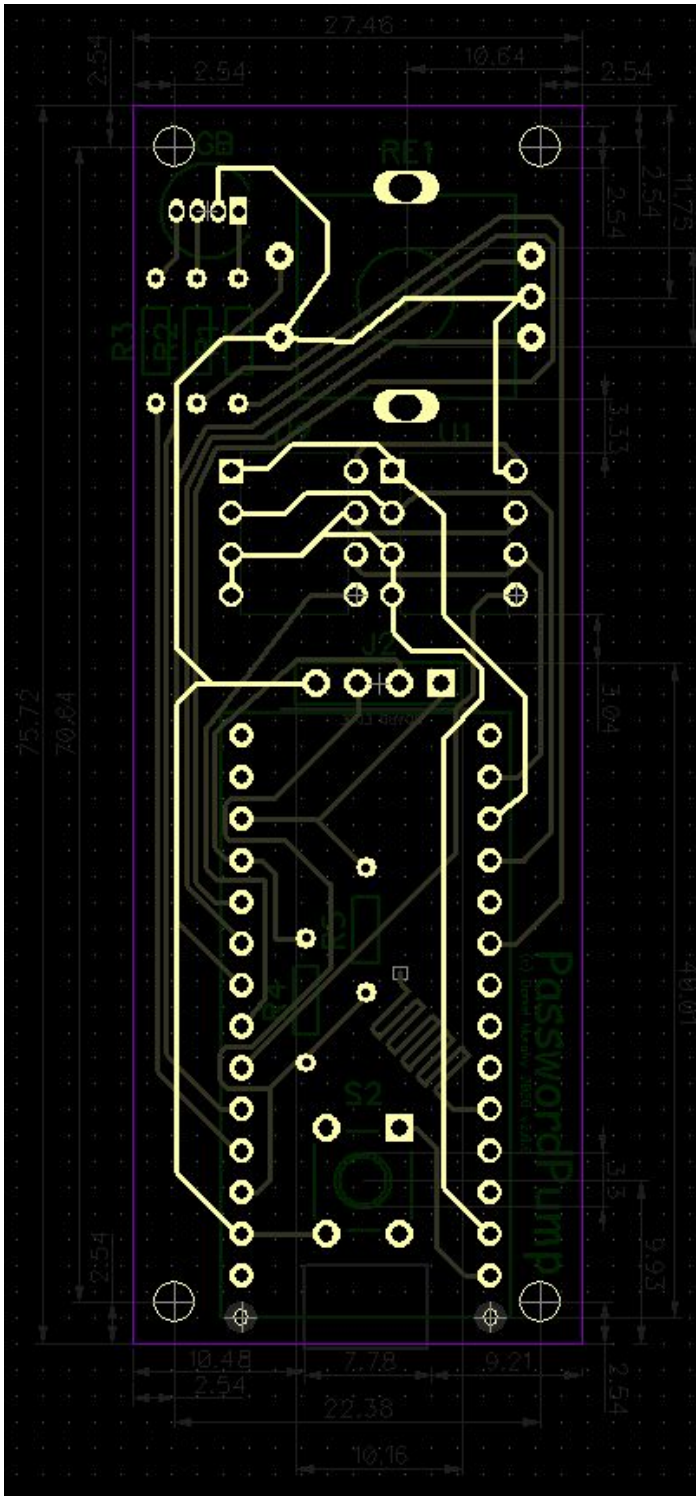
PCB

I'm using DIPTrace to do the PCB design. It's freely available. Download the DIPTrace file and open it with DIPTrace to edit the PCB design. This is really the first PCB I've ever designed, so if you have suggestions please feel free to let me know. I've also included the design files you'll need if you want to order the custom PCB on your own.

Top PCB Design



Bottom PCB Design



Connections

These are the connections made on the custom PCB, i.e. connections that must be made if you're building the project on a breadboard. See the Fritzing layout provided here if you're building this on a breadboard.

ItsyBitsy M4

Num	Name	Connect To / Notes
1	RS	reset button
2	3V	25LC512 Prim Pin 3 & 25LC512 Secondary Pin 3
3	AREF	
4	VHI	
5	A0	
6	A1	
7	A2	*connect to pin #9 for ItsyBitsy M0*
8	A3	
9	A4	220 Ohm resistor->RGB LED Pin 3
10	A5	
11	SCK	25LC512 Prim Pin 6 & 25LC512 Secondary Pin 6
12	MO	25LC512 Prim Pin 5 & 25LC512 Secondary Pin 5
13	MI	25LC512 Prim Pin 2 & 25LC512 Secondary Pin 2
14	2	
15	En	
16	swdio	
17	swclk	
18	3	
19	4	
20	RX	
21	TX	
22	SDA	SSD1306 SDA, 4.7k Ohm resistor->ItsyBitsy Pin 31
23	SCL	SSD1306 SCL, 4.7k Ohm resistor->ItsyBitsy Pin 31
24	5! (VHI Out)	220 Ohm resistor->RGB LED Pin 4
25	7	Rotary Encoder Pin 3
26	9	Rotary Encoder Pin 1
27	10	25LC512 Secondary Pin 1 Chip Select
28	11	25LC512 Primary Pin 1 Chip Select
29	12	Rotary Encoder Pin 4
30	13	220 Ohm resistor->RGB LED Pin 1
31	USB	SSD1306 VCC
32	G	RGB LED Pin 2, 25LC512 Prim & Secon Pin 4, SSD1306 Pin 1, Rotary Encoder Pins 2 & 5
33	BAT	

2 25LC512 (External EEprom)

Tested Part: MICROCHIP - 25LC512-I/P - 512K SPI™ Bus Serial EEPROM DIP8

25LC512 Primary

Num	Name	ConnectTo	Note
1	CS	pin 28 ItsyBitsy	Chip Select Input
2	SO	pin 13 ItsyBitsy	MISO - Serial Data Output
3	WP	pin 2 ItsyBitsy	Write Protect
4	Vss	pin 2 ItsyBitsy	Ground
5	SI	pin 12 ItsyBitsy	MOSI - Serial Data Input
6	SCK	pin 11 ItsyBitsy	SCLK - Serial Clock Input
7	HOLD	pin 2 ItsyBitsy	Hold Input
8	Vcc	pin 2 ItsyBitsy	Supply Voltage

25LC512 Secondary

Num	Name	ConnectTo	Note
1	CS	pin 27 ItsyBitsy	Chip Select Input
2	SO	pin 13 ItsyBitsy	MISO - Serial Data Output
3	WP	pin 2 ItsyBitsy	Write Protect
4	Vss	pin 2 ItsyBitsy	Ground
5	SI	pin 12 ItsyBitsy	MOSI - Serial Data Input
6	SCK	pin 11 ItsyBitsy	SCLK - Serial Clock Input
7	HOLD	pin 2 ItsyBitsy	Hold Input
8	Vcc	pin 2 ItsyBitsy	Supply Voltage

Rotary Encoder

1 2 3
4 5

Num	Name
1	ItsyBitsy Pin 26
2	ItsyBitsy Pin 32
3	ItsyBitsy Pin 25
4	ItsyBitsy Pin 29
5	ItsyBitsy Pin 32

SSD13306

GND VCC SCL SDA
1 2 3 4

Num	Name	ConnectTo

1	GND	ItsyBitsy Pin 32
2	VCC	ItsyBitsy Pin 31
3	SCL	ItsyBitsy Pin 23
4	SDA	ItsyBitsy Pin 22

RGB LED

Num	Name	ConnectTo
1	Red	220 Ohm resistor->ItsyBitsy Pin 30
2	Grnd	ItsyBitsy Pin 32
3	Green	220 Ohm resistor->ItsyBitsy Pin 9
4	Blue	220 Ohm resistor->ItsyBitsy Pin 24

A Note About the ItsyBitsy M0

As of v2.0.4 (of the custom PCB and the software) it is possible to substitute an Adafruit ItsyBitsy M**0** in place of the Adafruit ItsyBitsy M**4**. The reasons for doing this are purely economic; the M0 is $3 cheaper than the M4. The M0 operates at 48MHz and the M4 operates at 120MHz, but that difference is imperceptible to the user. The only difference that was significant for this project was that on the M4, pins A4 and A5 support PWM output, while instead on the M0 pins A1 and A2 are PWM capable. On the M4 pin A4 is used as PWM for green output for the RGB LED. For the M0 I had to move that to A2. Therefore, to keep things simple, I connected A2 and A4 on the custom PCB and adjusted the software with a few precompiler directives, so that for the M0 pin A2 serves as PWM output for green, and on the M4 pin A4 serves as PWM output for green. If you're building this project with the ItsyBitsy M0 please make adjustments accordingly and only use v2.0.4+ of the software (otherwise the RGB LED won't work correctly in all situations). Why didn't I just start with the ItsyBitsy M0? I had originally planned to use the M4's built in AES-256 crypto engine, but instead I used the RWeather library because I couldn't figure out how to get the built in version working. There is some code present as comments towards the end of the program if anyone with an M4 is interested in trying to get that working. The code compiles but it freezes the MCU when run. I have set aside a version of PasswordPump_v_2_0.ino that uses the built in crypto engine, if you're interested in trying to get it to work let me know and I'll send it to you.

Variable Costs

This is the cost of materials for each PasswordPump. The most expensive component is the MCU. The assembly takes about 45 minutes, and then there's the shipping costs to consider. If you have suggestions concerning how I could reduce costs, please let me know!

1 AdaFruit ItsyBitsy (32-bit ARM®, SAMD51 Cortex®-M4F MCU)*	$14.95
2 MICROCHIP - 25LC512-I/P - 512K SPI™ Bus Serial EEPROM DIP8	3.30
1 SSD1306 I2C LED display 128x32 pixels.	1.65
1 micro USB to USB cable 100cm	1.23
1 Custom PCB	1.00
1 Rotary Encoder	0.46
1 plastic knob for rotary encoder	0.58
2 IC DIP Sockets, 8 pins each	0.10
1 RGB LED diffused 5mm	0.03
3 220ohm resistors	0.01
2 4.7kohm resistors	0.01
Shipping Envelope	0.26
Solder	~0.10
Total Parts	$23.58

*Retail price from Adafruit

License