*MODLogger – Portable MODBUS TCP Data Recorder*

James M. Olson, OLSONJM

# Project Description

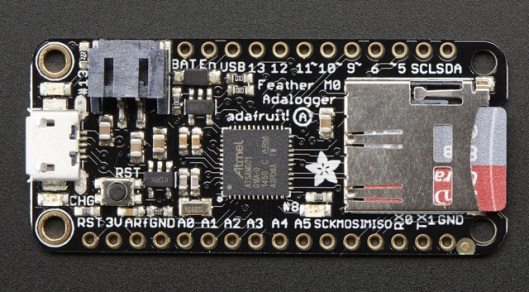
MODLogger is a stand-alone device which directly connects to external client devices via MODBUS TCP protocol for the purpose of downloading real time system data onto an SD card for the purpose of trending. The device provide a web interface from which a user can view the device settings, last measurement data, and download trend files. Device configuration can be easily changed to adapt to any MODBUS TCP enabled device by modifying the configuration file contained on the SD card.

# Project Features & Implementation

The MODLogger key features were developed based on the project requirement list that was proposed at the beginning of the project. Some additional features were also added during development to enhance the operation of the device. Selection of the hardware platform was founded on research into existing libraries which may provide a good base for project development. Additional factors in hardware selection included: ease of integration (modular), availability of parts, cost, and supporting libraries. The Adafruit Feather (Arduino base) platform provided a unique form factor which provided a powerful microcontroller (ARM Cortex M0) with 8x more FLASH and 16x more RAM than the ATMega 328 processor. The Feather provide a heavy-hitting processor but its small form factor has limited IO.

**Adafruit Feather M0 ‘Adalogger’**

<https://www.adafruit.com/products/2796>

  
Specifications

* Measures 2.0" x 0.9" x 0.28" (51mm x 23mm x 8mm) without headers soldered in
* Weight - 5.3 grams
* ATSAMD21G18 @ 48MHz with 3.3V logic/power
* 256KB of FLASH + 32KB of RAM
* No EEPROM
* 3.3V regulator with 500mA peak current output
* USB native support, comes with USB boot loader and serial port debugging
* 20 GPIO pins
* Hardware Serial, hardware I2C, hardware SPI support
* 8 x PWM pins
* 10 x analog inputs
* Built in 100mA lipoly charger with charging status indicator LED
* Pin #13 red LED for general purpose blinking
* Power/enable pin
* 4 mounting holes
* Reset button

Adafruit has other versions of the Feather which include WIFI, Bluetooth, or radio interface, but those features can also be added by plugging in add-on boards “wings”. The decision was made to develop the base hardware using the M0 board and add in wireless connectivity if time allowed.

Implementation of the features detailed in the original project requirement list are described below. Each requirement is described in following sections along with the hardware selected to implement the function.

**List 1: Project Requirements**

1. Connect to client MODBUS devices via RJ45 Ethernet and/or 2-wire RS485. (goal met)
2. Ability to download and store client data via MODBUS protocol. (goal met)
3. Store trend data to local SD card as plain ASCII text file. (goal met)
4. Integral web server that provides status of connection and the latest measurement data. (goal met)
5. Ability to change trend device settings. (goal met)
   1. Change static IP address
   2. Client device model selection.
6. Integrate at least 3 different MODBUS client device models. (goal exceeded)
7. Integral real time clock (RTC) for time stamping of measurements. (goal met)
8. Battery for momentary loss of power ride-through and controlled shutdown. (goal met)
9. Local display showing IP Address, battery status, and client connection status. (goal met)
10. Single measurement to occur on fixed time interval. (goal met)

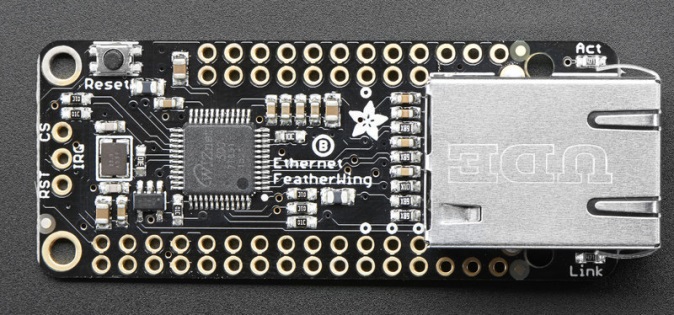
**A & D. MODBUS Slave Connection Method & Web Server**

There are two primary methods for connecting MODBUS devices: 1 – Serial RTU (RS485) and 2 – Ethernet TCP. The latter option was utilized for the project. Ethernet was selected because an Ethernet interfaces was already needed to provide web server, and because there was an existing Feather “Wing” Ethernet module available. However, the downside of this approach is that there is very limited documentation and libraries existing for MODBUS TCP. In fact, none were found to provide the interface required. Which meant considerable more programming and code development, but in the end a single ‘off-the-shelf’ module would provide the lower cost and easier to assemble modular approach that would meet another goal of the project. Serial MODBUS (RS485) can be implemented using the Maxim MAX485 IC which is commercially available from numerous commercial sources. Physical space was left during project development to allow for adding Serial MODBUS interface, and the code was developed to easily transition between interfaces.

The Adafruit Feather “Wing” Ethernet Module was selected for providing Ethernet web server interface and MODBUS TCP connect to slave. The module implements the WIZ5500 chipset which has been implemented in several Arduino boards with success.

**Adafruit Ethernet Feather Wing**

<https://www.adafruit.com/products/3201>



Specifications

* RJ45 Interface
* WIZ5500 Chipset
* SPI interface

The MODLogger can use a static IP or be provided an IP by DHCP server. As such, the connection method of the device into the network is quite flexible.











**B & C. MODBUS Slave Data storage**

The base microcontroller board included an SD card interface which provided means for storing collected data. The device was tested with various standard and high-capacity (SDHC) SD cards up to 32GB. Data is stored to the SD card in a CSV formatted file. This format was selected due to ease of importing to excel or future stand-alone analysis software. Excel provides the engine for converting the CSV data into graphs and trend plots.

**E & F. MODBUS Slave Device Models**

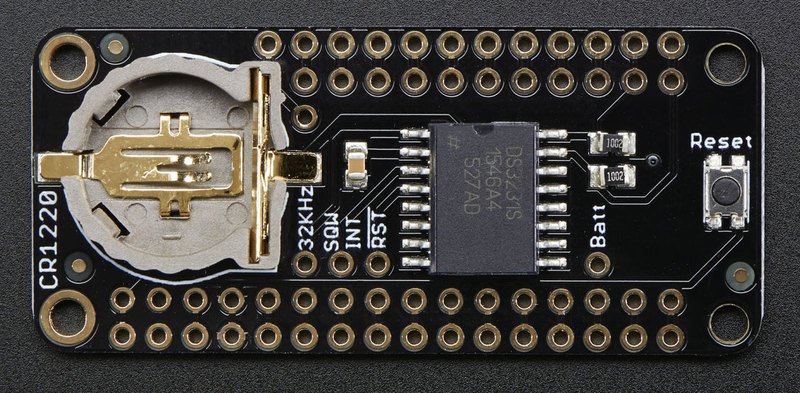
The MODLogger loads generic register data into memory during boot by reading a configuration file. The device code was developed to be device independent. As such, the goal of being able to communicate with at least 3 devices becomes irrelevant. Any device supporting MODBUS TCP protocol can be trending using MODLogger by entering the desired data registers into the configuration file. More detail on the development & formatting of the configuration file is provided in subsequent section of this report.

**G & J. Real Time Clock (RTC) for Time Stamping of Records**

Time synchronization is important with any logging device to make the data useful. This is especially true if comparing data from several trending devices where the only common variable is time. Adafruit conveniently had a RTC Feather Wing which can provide the needed time reference.

**Adafruit RTC Feather Wing**

<https://www.adafruit.com/products/3208>



Specifications

* Maxim DS3231 High Precision Real Time Clock (+/- 2 minutes per year drift)
* Temperature compensated crystal oscillator
* I2C Interface

The RTC wing provides a time standard for recording. However, the clock must still be initialized. The Ethernet interface provides a solution to this problem. A section in the configuration file was dedicated to providing an NTP server address. The device automatically synchronizes time with NTP server during boot if the RTC has lost its time reference (battery removed), and can also auto-resynchronize based on user defined time interval. A fallback time/date is hard coded in case the RTC fails and no NTP server is available.

**H. Backup Battery Power**

Battery power will ensure measurements and recording will continue for a short time if the external power lost. This may be important if trending line voltage conditions which may be lost and thus remove power supply to the MODLogger. An external UPS is one possible solution, but the Adafruit Adalogger M0 has an integrated Lipo battery charger circuit & DC connector which makes device battery connection seamless. The estimated power consumption was calculated at ~150mA. A 500mAh battery was selected to provide 1-2 hours record time with power lost.

**Adafruit 500mAH Lithium Ion battery**

<https://www.adafruit.com/products/1578>

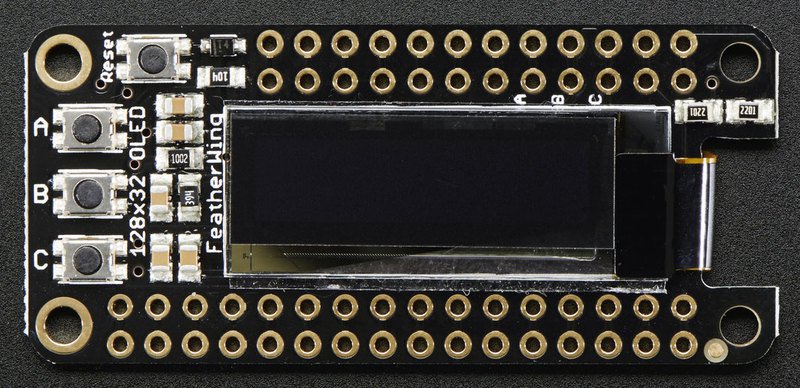


**I. Local Display**

OLED display was selected due to its high contrast and relatively low power consumption. Once again Adafruit has a modular solution for the Feather platform. The FeatherWing OLED module provides 128x32 pixel display which plugs directly on to the Feather base board. The display provides somewhat minimal graphic capabilities, but the intent of the data logger is to ‘set and forget’ for extended periods of time. As such, a complex local display is not required. The web interface provides much more information to the user, if desired. The module also has (3) built-in pushbuttons which were used for development. The buttons are tied to I/O pins which were used for the external buttons in the final assembly. As such, the code did not need to be changed, and the internal buttons remain functional if the front panel of the logger is removed.

**Adafruit OLED Wing**

<https://www.adafruit.com/products/2900>



Specifications

* 128x32 OLED Monochrome display
* I2C Interface
* 3 integral push buttons + reset

**ADDITIONAL HARDWARE**

1. PUSHBUTTONS - The onboard pushbuttons were mimicked on the MODLogger front panel with LED illuminated pushbuttons. The pushbuttons that were selected have separate pins for the momentary normally-open contact and the LED element. The LED element does not contain a current-limiting resistor. So, it was added on a separate proto board. Red and green buttons were obtained from Adafruit.

**Adafruit 16mm Illuminated Pushbutton – Momentary**

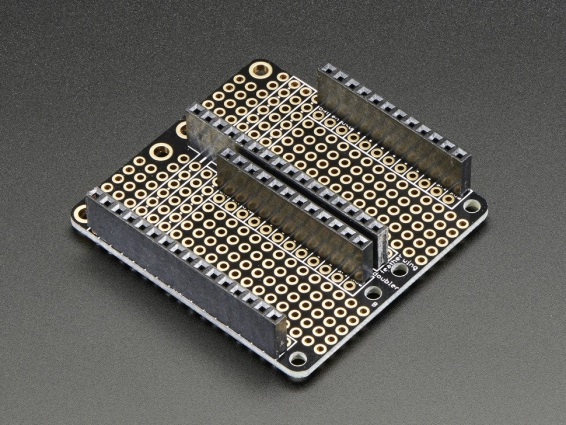
RED: <https://www.adafruit.com/products/1439> , GREEN: <https://www.adafruit.com/products/1443>

2. DOUBLER BOARD - The Adafruit feather boards contain a standard pin arrangement which allows for stacking of the devices. The devices come with standard header pins, but 12 & 16-pin stacking headers are available from Adafruit (<https://www.adafruit.com/products/2830>). It would have been possible to stack all the boards for this module, but it was decided to parallel stack to minimize the overall height of the device. Adafruit sells a backplane board for exactly that purpose. The ‘doubler’ board allows for side-by-side stacking while maintaining the parallel arrangement of the 12 and 16-pin headers.

**Adafruit FeatherWing Doubler**

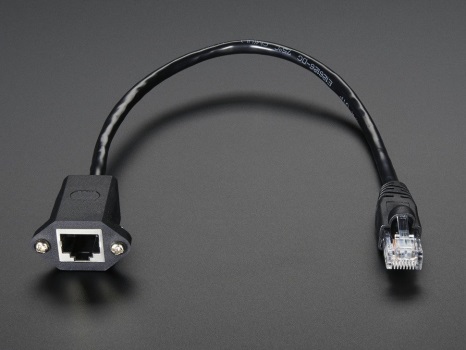
<https://www.adafruit.com/products/2890>



3. PANEL MOUNT EXTENSION/ADAPTER – A panel mount adapter/extender was added to bring the Ethernet and USB plugs to the exterior of the case. Both USB and RJ45 cables are available from Adafruit. The USB adapter changed the USB Micro connector on the Adalogger board to USB-B. This interface was also used for programming the device.

USB: <https://www.adafruit.com/products/937>

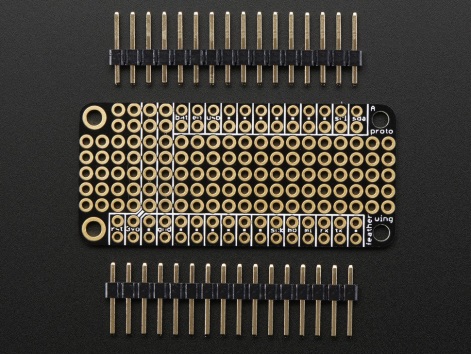
Ethernet: <https://www.adafruit.com/products/909>

4. FEATHER PROTOTYPING BOARD - A FeatherWing prototyping board was used to connect additional I/O pins to the micro-controller and to provide interface to external push-buttons. Additional detail on how this board was wired is provided later in this report.

**Adafruit FeatherWing Proto**

<https://www.adafruit.com/products/2884>



# Project Development

# A. Initial Development & Testing

Initial development was begun by following the tutorials on Adafruit for gaining communication with the individual FeatherWing modules. It was also discovered that the latest Arduino IDE v1.7 was not compatible with the Feather because Arduino dropped support for 3rd party boards. Therefore, Arduino IDE version 1.6.11 was installed and used through the entirety of the project to develop the code for the MODLogger.

Arduino v1.6.11 Installation Link: <https://www.arduino.cc/download_handler.php?f=/arduino-1.6.11-windows.zip>

Some additional libraries also must be installed into Arduino IDE:

Version 1.6.11 must add in supported for Adafruit SAMD boards.  File->Preferences->Addional Boards Manager URLs: <https://adafruit.github.io/arduino-board-index/package_adafruit_index.json> .Then go to Tools ->Boards->Board Manager and install 'Adafruit SAMD Boards by Adafruit'. Version 1.0.13 used for this project.

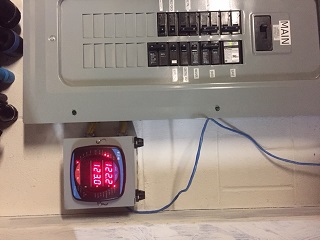
Must install Adafruit Feather USB drivers (does not use std FDTI chipset) before connecting board to USB port:<https://github.com/adafruit/Adafruit_Windows_Drivers/releases/download/1.0.0.0/adafruit_drivers.exe>

**FeatherWing Board Support**

* Ethernet Module
  + Support Library: <https://github.com/adafruit/Ethernet2/archive/master.zip>
  + Tutorial: <https://learn.adafruit.com/adafruit-wiz5500-wiznet-ethernet-featherwing/overview>
* Real Time Clock Module
  + Support Library: <https://github.com/adafruit/RTClib/archive/master.zip>
  + Tutorial: <https://learn.adafruit.com/ds3231-precision-rtc-featherwing/overview>
* OLED Graphic Display Module
  + Support Library 1: <https://github.com/adafruit/Adafruit_SSD1306/archive/master.zip>
  + Support Library 2: <https://github.com/adafruit/Adafruit-GFX-Library/archive/master.zip>
  + Support Library 3: <https://github.com/adafruit/Adafruit_FeatherOLED/archive/master.zip>
  + Tutorial: <https://learn.adafruit.com/adafruit-oled-featherwing/featheroled-library?view=all#overview>

A MODBUS TCP slave device was going to be needed for testing communications and trending. Two Eaton PowerXpert 2000 series power quality meters were obtained for used on this project. The PX meter is a 3-phase (or 1-phase) volt, amp, power meter. One meter was installed on home residential panel and second meter was connected to a power cord. The home meter was wired with 200/5 current transformers connected to load feed to a 90A sub-panel, and the voltage leads connected with A&B phases on the two 120V legs of the 240V service (referenced to neutral). The second meter was wired to a wall plug with A&B phases connected to 120V outlet and referenced to neutral wire. The PX meter has an integral Ethernet RJ45 port which can be used to serve a web page and/or MODBUS TCP data feed (all volt, amp, power measurements). The IP address of the meter was set to a static address of 192.168.1.100. The second ‘portable’ meter was used for development in class, or when away from home. The MODBUS TCP register list for the Eaton PX2000 meter is readily available on-line at:

<http://www.eaton.com/ecm/idcplg?IdcService=GET_FILE&allowInterrupt=1&RevisionSelectionMethod=LatestReleased&noSaveAs=0&Rendition=Primary&dDocName=CT_265774>

****

Eaton PX2000 Meter on House Panel (displaying A-N and B-N voltage)

# B. Revision 0 – MODBUS TCP connection to Slave

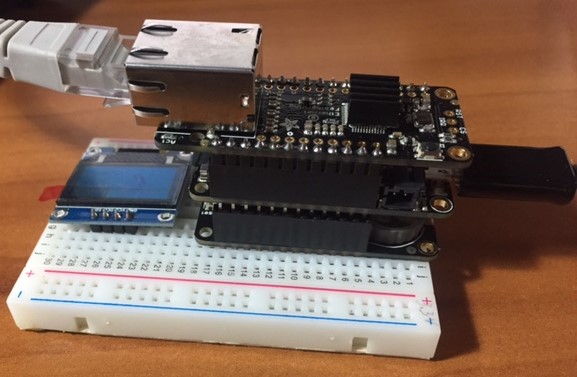
The goal of the first code release was to get MODBUS communications working to slave device, time stamp measurement with RTC, and push the slave data out to the serial debug port.

Some research turned up a MODBUS TCP library which was initially designed for connecting two Arduino’s together and letting them mirror data registers back and forth (<http://myarduinoprojects.com/modbus.html>) . The CPP library file was downloaded and reviewed and it seemed like a good starting point for building a Master-Slave MODBUS library. A few minor changes were necessary to the library to get it work reliably for this project:

* Library was published with a fixed master/slave static IP address hard coded into the library file. The library was modified so that slave address could be any IP4 (4-byte) address.
* The library did not provide any delay to allow master time to connect to slave device over IP which would cause intermittent failed connections. Added 10ms delay after initial port connection (MODBUS port 502). Eliminated connection problem.
* Added return value to MB Master data request to slave. If connection fails, the request function returns false to let main program know that connection was unsuccessful.
* The read port function in the library would sometimes hang if connection was slow or lost mid-transmission. Added a read counter to the loop to terminate the port read command if connection was lost.
* Changed library support from using Ethernet.h to Ethernet2.h. The latter is required when using the Feather platform vs. standard Arduino with Ethernet shield.

Once these modifications were performed connection was successful to Eaton PX meter using MODBUS TCP protocol. The A-N and B-N voltages were read from the PX meter using MODBUS read holding register function code (03) to registers 4010 and 4012. Reading floating point values from a MODBUS device is not quite as simple as reading the number directly. Most devices store data in 16-bit storage registers. So, to read a 32-bit floating point number you must read the two 16-bit registers, flip the LSB and MSB (for most MODBUS devices), combine into a 32-bit word, and then convert the 32-bit word into a float point number (using memcpy() in C). Not all MODBUS devices flip the upper and lower 16-bit registers for floating point numbers. This conditional state was handled in the configuration file (later revision).

REV 0 device with RTC and EthernetWing.

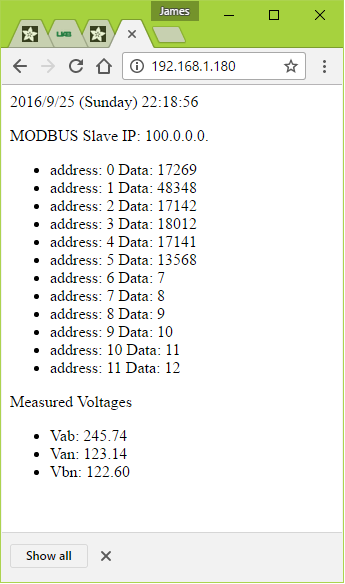


Communication was successful with slave device:



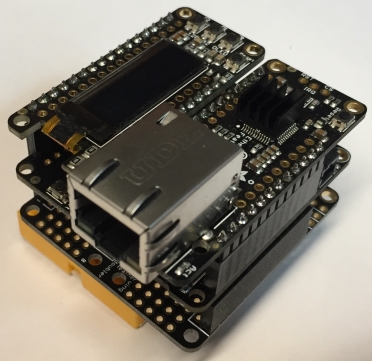
The Master request and slave response (xx.xx.xx…) was printed from MsgMODBUS library using serial debug interface.

Basic web server function of the EthernetWing was enabled over port 80 and data was pushed via basic txt/html. Web formatting verified on both IE and Chrome.



# C. Revision 1 – OLED Display and SD Card

The first revision of the base code integrated the SD card and the OLED display. The Feather doubler board was used for this hardware revision.



OLED Display   
  
RTC Module  
  
Ethernet Module  
  
Adalogger M0 Microcontroller  
  
Feather Doubler Board

SD Card Read/Write

It was determined quickly during coding that if the intent of the device was to be able to connect to more than on type of slave device, a configuration file would need to be created. There are no standards when it comes to register arrangement in devices. As such, the same data point (say voltage phase a to b) could be located in just about any location in slave memory. The configuration file was set such that the user can define many of the operational characteristics of the MODLogger and configure the OLED display screens. The configuration file was named ‘modlog.cfg’ and was placed in the ‘/config’ directory on the SD card. The code scans for this configuration file during boot, and will not run without it (will show error screen).

*MODLOG.CFG*

[DEVICE]  
MAC=0x90.0xA2.0xDA.0x0E.0x94.0xB5  
IP=0.0.0.0  
SIP=24.178.187.21  
GW=192.168.1.1  
SN=255.0.0.0  
ID=PXM2000  
MEASURE=10  
RECORD=60  
SCREEN=60  
TIMESERVER=time.nist.gov  
AUTOTS=1  
TSINTERVAL=24  
[END\_DEVICE]

:Format = "Tag Name",Flip LSB (1=true), MODBUS Function Code, Address (DEC), # of registers  
[REGISTERS]  
LL Voltages,0,0,0,0,NA   
Vab,1,3,4002,2,V  
Vbc,1,3,4004,2,V  
Vca,1,3,4006,2,V  
LN Voltages,0,0,0,0,NA  
Van,1,3,4010,2,V  
Vbn,1,3,4012,2,V  
Vcn,1,3,4014,2,V  
Line Currents,0,0,0,0,NA  
Ia,1,3,5002,2,A  
Ib,1,3,5004,2,A  
Ic,1,3,5006,2,A  
Averages,0,0,0,0,NA  
Vll,1,3,4016,2,V  
Vln,1,3,4016,2,V  
Iavg,1,3,5012,2,A  
System Power 1,0,0,0,0,NA  
P3,1,3,6006,2,W  
S3,1,3,6070,2,VA  
Q3,1,3,6134,2,VAR  
System Power 2,0,0,0,0,NA  
pf,1,3,6220,2,NA  
hz,1,3,11000,2,NA  
blank,0,1,0,0,NA  
[END\_REGISTERS]

Description of configuration file

1. The first section of the configuration file contains the device configuration.

* MAC = MAC address of the EthernetWing module (sticker included on module during purchase)
* IP = Static IP address of device (set to 0.0.0.0 to use DHCP 🡨 functionality added in next revision)
* SIP = MODBUS Slave device IP address. This is device from which you are polling data.
* GW = Default gateway (used if using static IP address)
* SN = Subnet mask (used if using static IP address)
* ID = ‘Name’ of the slave device. This will appear in log files and on the MODLogger OLED screen.
* MEASURE = Slave polling time interval in seconds.
* RECORD = How often to save measurement to SD card.
* SCREEN = OLED display screen blanking timeout.
* TIMESERVER = NTP time server for RTC synchronization.
* AUTOTS = Automatic time synchronization feature 1 = ENABLE, 0 = DISABLE.
* TSINTERVAL = Time interval for automatic time synchronization in hours (if enabled).

2. The second section of the configuration file defines the slave registers and sets up the local display on the MODLogger. Each line contains (6) five CSV values which determine what the main program should do with the data on the line (i.e. VALUE 1, VALUE 2, VALUE 3, VALUE 4, VALUE 5, VALUE 6)

* VALUE 1 = Name of the Data point
* VALUE 2 = Should the program flip the LSB and MSB if the data point spans two 16-bit registers. 1 = YES, 0 = NO
* VALUE 3 = MODBUS Function code

Supported Function Codes

* + - 1 = READ COILS
    - 2 = READ DISCRETE INPUT
    - 3 = READ REGISTERS
    - 4 = READ INPUT REGSITER
* VALUE 4 = Decimal Register address in the slave where data is being held.
* VALUE 5 = Number of registers to read (if data spans more than one address).
* VALUE 6 = Units for the data point (for function codes 3 & 4). If value is set to ‘NA’, no units will be added to value on OLED or WEB. Units will be added in [] after node name in CSV data log file title line.

There are some special formatting options for the register section of the code which can be used to define the OLED display screens. Each screen on the OLED spans 4 registers on the list. This is fixed in code. As such, the registers must be listed in the logical order you want them to appear on the OLED screen. Special entries can be made to provide ‘Title’ lines on the display or ‘blank’ lines to get data arranged as the user wishes.

* Title lines
  + ‘Title’ is entered into the Register Name (VALUE 1)
  + A ‘0’ zero is entered as the MODBUS function code (VALUE 3)
  + A ‘0’ zero is entered into the MODBUS register column (VALUE 4)
  + All other values do not matter.
* Blank lines
  + A ‘1’ one is entered as the MODBUS function code (VALUE 3)
  + A ‘0’ zero is entered into the MODBUS register column (VALUE 4)

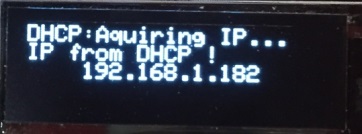
OLED Screen Displays

The intent of the local display is to provide basic information to user. Additional information will be made available from the web display. There are three generic screens that are typical for all slave devices, the remaining screens are slave data specific as define in the configuration file. There are also a few boot screens and error screens that were created to inform user of status of the device during boot.

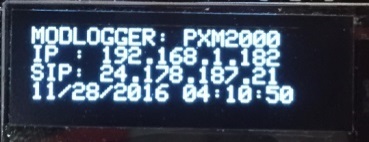
Boot Screen 1 – Welcome screen showing the device name and version number.



Boot Screen 2 – This screen is displayed if DHCP is used for assigning an IP address. If static IP is used this screen is skipped and the device will go direct to Default Screen 1.



Default Screen 1 - Displays the Slave device name, the Slave IP Address, the MODLOgger IP address, and the time as read from the RTC. Note all times are in UTC.



Default Screen 2 – Displays the status of the SD card, notes whether logging is enabled or disabled, the current log filename, and the number of measurements logged to the current record file.



Default Screen 3 – Displays the current power source (battery or external DC), the battery/charging circuit voltage, and the external DC-in (USB) voltage.



Error screens will display if RTC fails or SD card configuration file is not detected and the MODLogger will halt operation. Configuration file is required for operation. If it is determined that the RTC has lost synchronization the display will let you know it is attempting to synchronize the clock.

Sample Slave Data Screen – Screens will vary based on the configuration defined in the modlog.cfg file.



# D. Revision 2 – Final Configuration

The next revision of the code included completing the web view and formatting of the SD datalogger files.

Web server html was programmed to show MODLogger device data, slave data, and to create download links to the files on the SD card. HTML page is severed up as plain text/html.

The filenames for the data logging were set to be created with random unique names starting with “LF” followed by a random 5 digit number, and finally a character ‘A-Z’. The program code checks the SD card to make sure it doesn’t already exist before creating the new log file. The first line of the data file is a title line which lists the device name followed by the titles given to all the slave data points.

The green pushbutton advances the OLED screen, and the LED is programmed to flash when transmitting data to the TCP slave. The red pushbutton illuminates when successful communication is established with the Slave, and when pressed enables/disables logging.

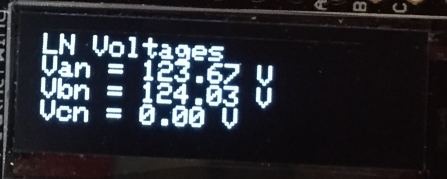
# Final Results

Final design shows the implemented web view for the PX2000 test meter, a sample CSV log file, and the data imported into excel and plotted into a data trend. The OLED screens are also shown for the different views created by the configuration file (same file as shown in previous section). Overall results exceeded expectations.

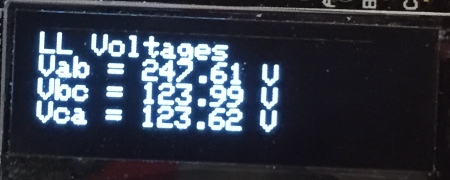
**A. OLED Slave Device Specific Screens**

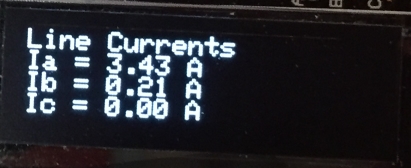
Screen 1 – Device Name and Slave IP Address. Using DHCP

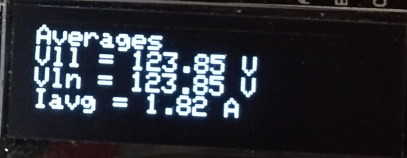
**** Device Name Imported from configuration file.

Screen 2 – Device Specific

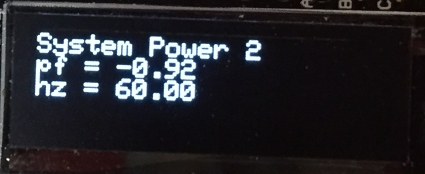
Screen title line (no data) from configuration file

Screen 3 – Device Specific

Screen 4 – Device Specific  


Screen 5 – Device Specific

Screen 6 – Device Specific

Screen 7 – Device Specific

**B. CSV Log File (Auto-created by program)**

Logging was enabled on home meter for couple days and trend file was extracted. Only the first few lines of the log file are shown.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PXM2000 | Vab[V] | Vbc[V] | Vca[V] | Van[V] | Vbn[V] | Vcn[V] | Ia[A] | Ib[A] | Ic[A] | Vll[V] | Vln[V] | Iavg[A] | P3[W] | S3[VA] | Q3[VAR] | pf | hz |
| 1.48E+09 | 247.07 | 123.45 | 123.63 | 123.6 | 123.52 | 0 | 3.47 | 0.21 | 0 | 123.56 | 123.56 | 1.84 | 415.08 | 451.84 | -141.66 | -0.92 | 60 |
| 1.48E+09 | 247.15 | 123.53 | 123.63 | 123.61 | 123.54 | 0 | 3.46 | 0.21 | 0 | 123.58 | 123.58 | 1.83 | 414.09 | 454.84 | -143.61 | -0.92 | 59.98 |

Note the time is shown in unix time format (seconds since January 1, 1970). Time column can easily be converted to standard time format by using excel formula: CELL#/(60\*60\*24)+”1/1/1970” where CELL# is the unix time.

**C. CSV Log Data Imported to Excel**

Logging for power imported into an excel XY scatter plot

Logging for AB line voltage imported into an excel XY scatter plot

Similar plots can be created for all data points.

**D. Web View for PX2000 Meter**

Web view is generated in plain-text HTML as shown below…

# Modlogger - PXM2000

Device Time (UTC): Monday 11/28/2016 05:44:00

Last Measurement (UTC): Monday 11/28/2016 05:43:56

Measurement Interval: 10 second(s)

Record Interval: 60 second(s)

Display Timeout: 60 second(s)

SD Card: INSTALLED

Current SD Log File Name: LF95902O.csv

Number of Data Points Stored to currend Log File: 92

Configuration File: [MODLOG.CFG](http://192.168.1.182/config/modlog.cfg)

MODBUS Slave IP: 24.178.187.21

MODBUS Slave Status: CONNECTED

## Power Supply Data

Current Power Source: External DC

Battery/Charger Voltage: 4.38VDC

External DC Voltage: 5.13VDC

## Data Points from MODBUS Slave

### LL Voltages

Vab = 248.35 V

Vbc = 124.14 V

Vca = 124.22 V

### LN Voltages

Van = 124.24 V

Vbn = 124.15 V

Vcn = 0.00 V

### Line Currents

Ia = 3.44 A

Ib = 3.60 A

Ic = 0.00 A

### Averages

Vll = 124.20 V

Vln = 124.20 V

Iavg = 3.52 A

### System Power 1

P3 = 834.37 kW

S3 = 876.82 kVA

Q3 = -33.31 kVAR

### System Power 2

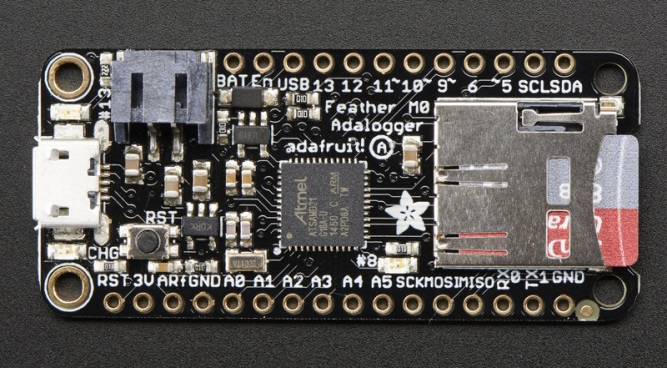
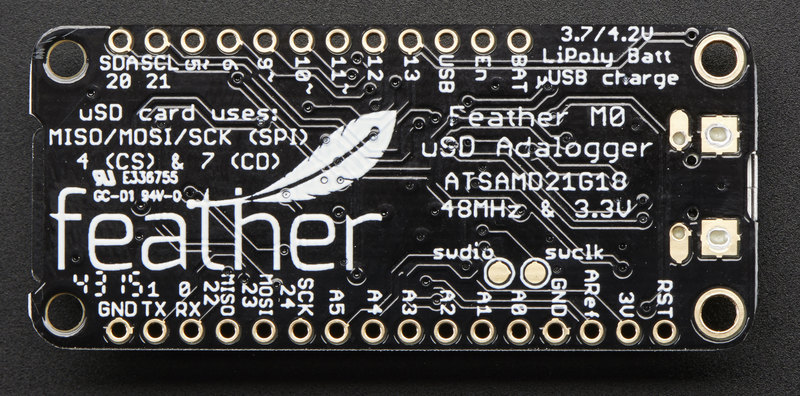
pf = -0.95

hz = 60.01

## SD Card Files - Click Filename to Download

|  |  |
| --- | --- |
| **Bytes** | **Filename** |
| 314 | [LF11173R.CSV](http://192.168.1.182/LF11173R.CSV) |
| 2213 | [LF67552K.CSV](http://192.168.1.182/LF67552K.CSV) |
| 1166 | [LF71530Y.CSV](http://192.168.1.182/LF71530Y.CSV) |
| 265 | [LF35545H.CSV](http://192.168.1.182/LF35545H.CSV) |
| 268 | [LF79165M.CSV](http://192.168.1.182/LF79165M.CSV) |
| 502 | [LF99559P.CSV](http://192.168.1.182/LF99559P.CSV) |
| 1844 | [LF75187X.CSV](http://192.168.1.182/LF75187X.CSV) |
| 603 | [LF7195D.CSV](http://192.168.1.182/LF7195D.CSV) |
| 487 | [LF91924A.CSV](http://192.168.1.182/LF91924A.CSV) |
| 10679 | [LF27910T.CSV](http://192.168.1.182/LF27910T.CSV) |
| 1188 | [LF52281J.CSV](http://192.168.1.182/LF52281J.CSV) |
| 1065 | [LF31567S.CSV](http://192.168.1.182/LF31567S.CSV) |
| 361 | [LF55938I.CSV](http://192.168.1.182/LF55938I.CSV) |
| 10997 | [LF95902O.CSV](http://192.168.1.182/LF95902O.CSV) |

Adafruit M0 Adalogger – Pin Utilization

Pin Utilization & Description

| Pin ID | Description | Main  Board | RTC  Module | Ethernet  Module | OLED  Module | Proto  Board |
| --- | --- | --- | --- | --- | --- | --- |
| BAT | + Terminal from Battery Plug | X |  |  |  |  |
| EN | 3.3V Regulator Enable Pin (Pulled up to 3.3V). Ground to disable on-board regulator. | X |  |  |  |  |
| USB | + Voltage to/from Micro-USB plug |  |  |  |  | X |
| 13 | GPIO #13 (also RED LED on base board) | X |  |  |  |  |
| 12 | GPIO #12 |  |  |  |  | X  Pnl LED |
| 11 | GPIO #11 |  |  |  |  | X  Pnl LED |
| 10 | GPIO #10 |  |  | X (CS – SPI) |  |  |
| 9\* | GPIO #9 & Analog in A7 connected to divider to measure Lipo battery. (sits around 2VDC due to divider). | X |  |  | X  Button A |  |
| 6 | GPIO #6 |  |  |  | X  Button B | X  Pnl Btn |
| 5 | GPIO #5 |  |  |  | X  Button C | X  Pnl Btn |
| SCL  (21) | I2C Clock Pin (no internal pull-up) |  | X (0x68)  10k PU |  | X (0x38)  2.2k PU |  |
| SDA  (20) | I2C Data Pin (no internal pull-up) |  | X  (0x68)  10k PU |  | X (0X38)  2.2k PU |  |
| RST | Reset Pin. Ground to reset device |  | X | X | X |  |
| 3V | Output from 3.3V regulator 500mA maximum | X | X | X | X |  |
| Aref | Analog Reverence (3.3V Maximum) |  |  |  |  |  |
| GND | System Ground Plane | X | X | X | X | X |
| A0 | True Analog output (DAC) or Analog input |  |  |  |  |  |
| A1 | Analog Input or Digital I/O |  |  |  |  | X (USB V) |
| A2 | Analog Input or Digital I/O |  |  |  |  |  |
| A3 | Analog Input or Digital I/O |  |  |  |  |  |
| A4 | Analog Input or Digital I/O |  |  |  |  |  |
| A5 | Analog Input or Digital I/O |  |  |  |  |  |
| SCK  (24) | SPI bus Clock | X |  | X |  |  |
| MOSI | SPI data bus | X |  | X |  |  |
| MISO | SPI data bus | X |  | X |  |  |
| 0(RX) | Serial Bus Receive – GPIO #0 or Analog In  Serial1 (hardware UART) |  |  |  |  |  |
| 1(TX) | Serial Bus Transmit – GPIO #1 or Analog In |  |  |  |  |  |

\* Conflict at GPIO #9. Button A cannot be used on OLED display.

Internal Adalogger M0 Pin Utilization & Description (not brought out to headers)

| Pin ID | Description | Main  Board | RTC  Module | Ethernet  Module | OLED  Module |
| --- | --- | --- | --- | --- | --- |
| 4 | GPIO #4 – Used for Chip Select on SD Card SPI Bus | X (CS) |  |  |  |
| 7 | GPIO #7 - Used as Card detect for SD Card. Input is pulled low when card is removed. | X |  |  |  |
| 8 | GPIO #8 – Wired to greed LED on base board | X |  |  |  |

Power Consumption

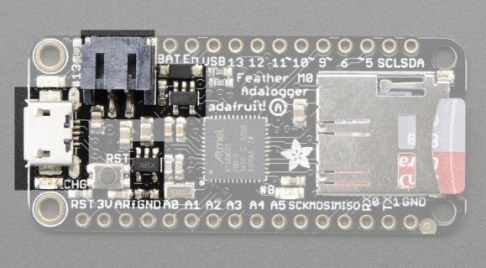
Adafruit Adalogger M0 regulator maximum output +500.0 mA

* RTC Board (Onboard CR1220 12mm Coin Cell ~5yr lifespan) -000.2 mA
* OLED Board (maximum current draw) -010.0 mA
* Ethernet Module (maximum during transmit) -150.0 mA

Regulator balance (overhead) 339.8 mA

Actual current draw after battery fully charged was measured to be ~130 mA. Below projected total of 160 mA. This was measured using the Eversame Digital USB Power Meter.

Battery Charger



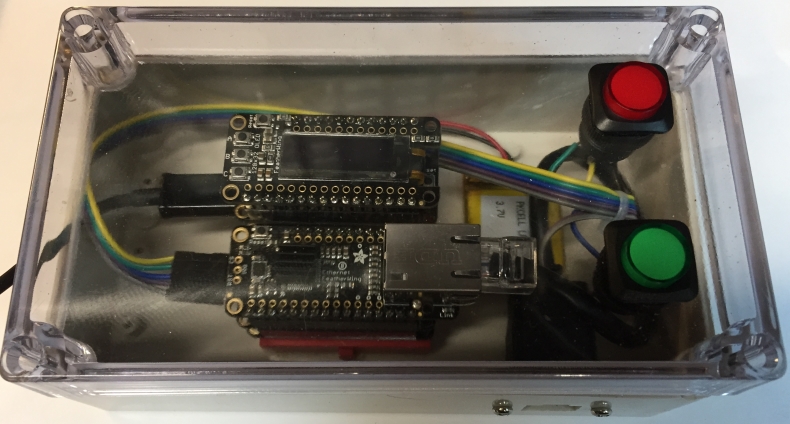
Onboard battery Lipoly Battery charger will supply 100mA to battery when 5V USB plug is connected. Fully charged battery will measure around 4.2V and the will drop to around 3.7V for majority of its run cycle. Battery voltage will drop to around 3.2V when it is about to flat-line.

Project Costs

Total project material cost: $113.72

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Quantity | Cost | Part# | Supplier | Description | Link | Line Total |
| 1 | $21.95 | 2796 | [www.adafruit.com](http://www.adafruit.com/) | Adafruit Feather M0 Adalogger | [www.adafruit.com/products/2796](http://www.adafruit.com/products/2796) | $21.95 |
| 1 | $19.95 | 3201 | [www.adafruit.com](http://www.adafruit.com/) | Adafruit Feather Ethernet Wing | [www.adafruit.com/products/3201](http://www.adafruit.com/products/3201) | $19.95 |
| 1 | $13.95 | 3028 | [www.adafruit.com](http://www.adafruit.com/) | Adafruit DS3231 Precision RTC | [www.adafruit.com/products/3028](http://www.adafruit.com/products/3028) | $13.95 |
| 1 | $14.95 | 2900 | [www.adafruit.com](http://www.adafruit.com/) | Adafruit 128x32 OLED Display | [www.adafruit.com/products/2900](http://www.adafruit.com/products/2900) | $14.95 |
| 1 | $5.95 | 1570 | [www.adafruit.com](http://www.adafruit.com/) | Lithium Ion 3.7V 500mAh Battery | [www.adafruit.com/products/1570](http://www.adafruit.com/products/1570) | $5.95 |
| 1 | $7.50 | 2890 | [www.adafruit.com](http://www.adafruit.com/) | Adafruit Feather Doubler | [www.adafruit.com/products/2890](http://www.adafruit.com/products/2890) | $7.50 |
| 1 | $12.62 | B0143YVNHW | [www.amazon.com](http://www.amazon.com/) | Enclosure | [www.amazon.com/gp/product/B0143YVNHW/](http://www.amazon.com/gp/product/B0143YVNHW/) | $12.62 |
| 1 | $4.95 | 909 | [www.adafruit.com](http://www.adafruit.com/) | Ethernet Extender | [www.adafruit.com/products/909](http://www.adafruit.com/products/909) | $4.95 |
| 1 | $3.95 | 937 | [www.adafruit.com](http://www.adafruit.com/) | USB Extender | [www.adafruit.com/products/937](http://www.adafruit.com/products/937) | $3.95 |
| 1 | $1.50 | 1439 | [www.adafruit.com](http://www.adafruit.com/) | Pushbutton - RED LED | [www.adafruit.com/products/1439](http://www.adafruit.com/products/1439) | $1.50 |
| 1 | $1.50 | 1440 | [www.adafruit.com](http://www.adafruit.com/) | Pushbutton -Green LED | [www.adafruit.com/products/1440](http://www.adafruit.com/products/1440) | $1.50 |
| 1 | $4.95 | 2884 | [www.adafruit.com](http://www.adafruit.com/) | Feather Potor Board | [www.adafruit.com/products/2884](http://www.adafruit.com/products/2884) | $4.95 |

Final Assembly

Stack 1 (on Doubler Board)

1 – (bottom) AdaLogger M0  
2 – (middle) RTC Board  
3 – (top) OLED Board

Stack 2 (on Doubler Board)

1 – (bottom) ProtoBoard  
2 – (top) EthernetWing

500mAH Battery

Ethernet Panel Plug

# Future Development

Some basic improvements could be made to the user interface and hardware to further expand the capabilities of the device.

1. Both RS485 serial and RJ45 interface as standard (MODBUS serial RTU & TCP). With software selection of desired interface.
2. Second module which can be used to provide protocol translation to client devices which do not inherently communicate via MODBUS (i.e. CAN, SPI, I2C).
3. Bluetooth interface (disable/enable via software) and phone application for viewing data.
4. Automatic MIN, MAX, and AVERAGE calculation for each time interval measurement.
5. Event triggers for forced time sampling (i.e. trigger on high current or low voltage).
6. Web interface can be greatly improved using client side scripting or general visual improvements.
7. Enclosure cost & size could be reduced for final assembly.

# Additional References

MODBUS TCP request and packet formatting: <http://www.simplymodbus.ca/TCP.htm>

Introduction to MODBUS TCP Protocol: <http://www.prosoft-technology.com/kb/assets/intro_modbustcp.pdf>

MODSCAN Software (used for protocol debugging): <http://www.win-tech.com/html/modscan32.htm>

MgsMODBUS Library: <http://myarduinoprojects.com/modbus.html>

Eversame Digital USB Power Meter: <https://www.amazon.com/Eversame-Multimeter-Chargers-Capacity-Banks-Black/dp/B01D9Y6ZFW>

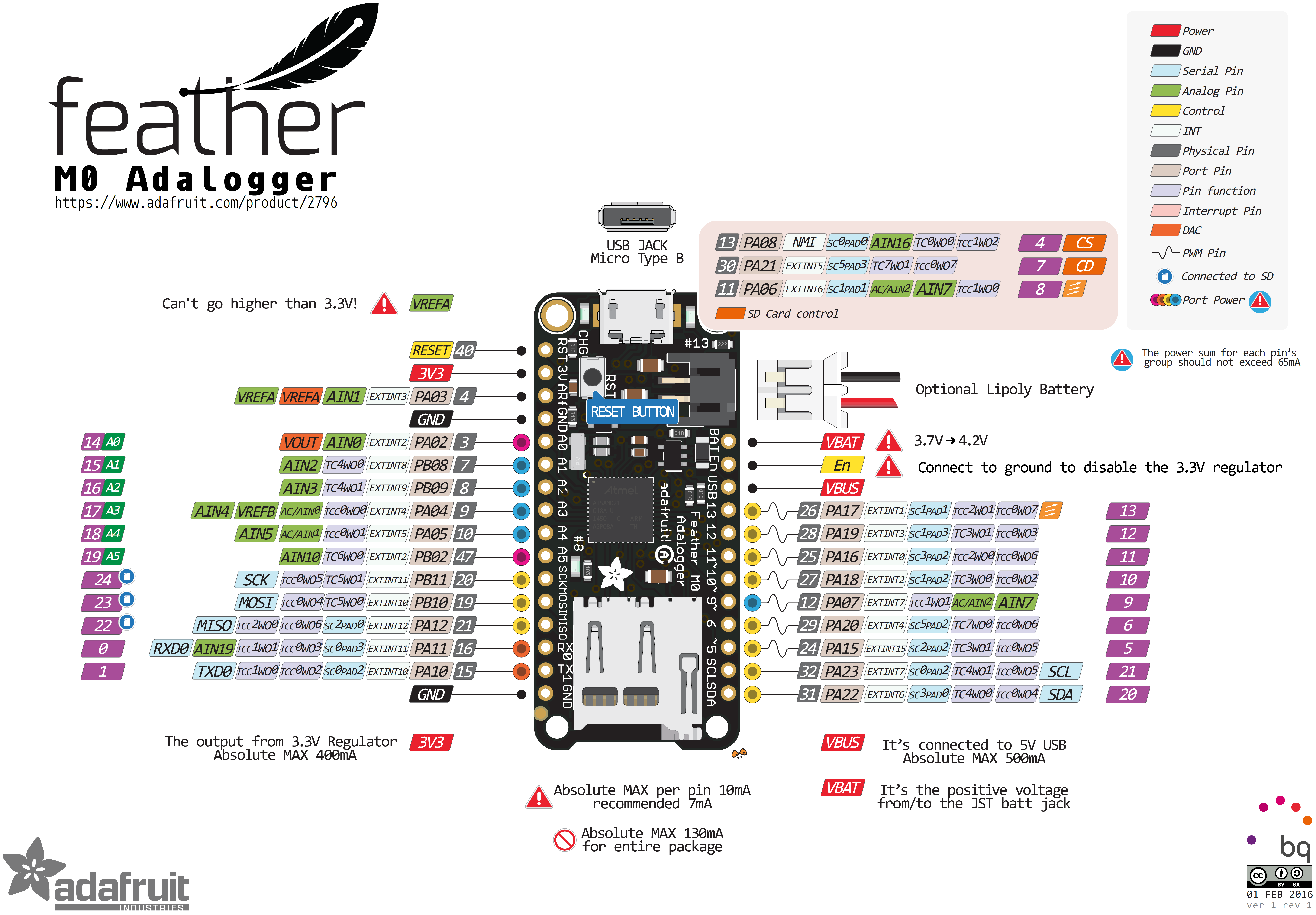
# What Was Learned

Project extended students general knowledge in hardware/software integration. Specific experience was gained in following areas:

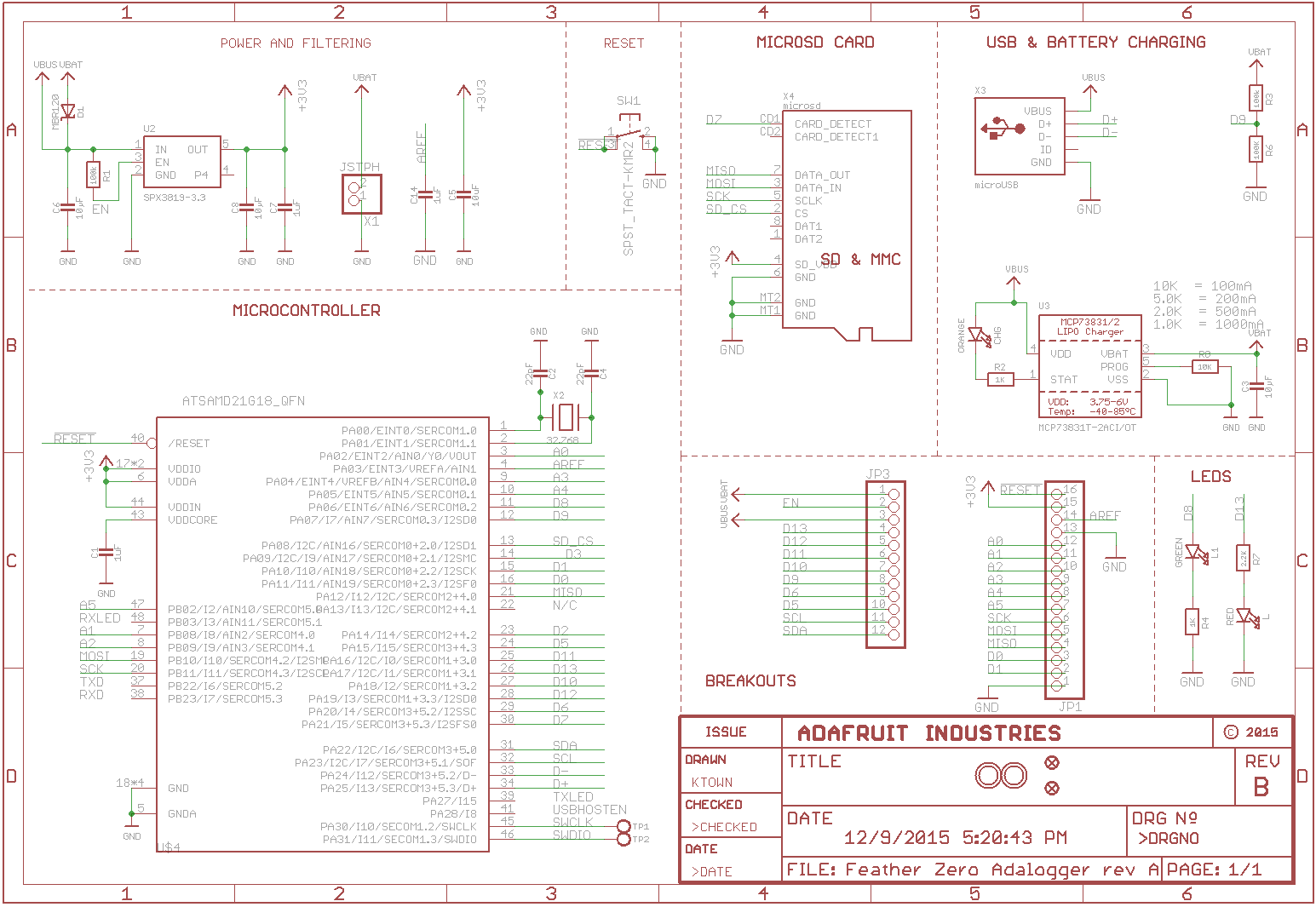
1. MODBUS TCP protocols – Message request/response formatting for TCP.  
2. MODBUS register data post-processing.  
3. Ethernet package byte formatting.  
4. New Hardware platform: Adafruit Feather  
5. Connecting to SD card using SPI and file read/write.  
6. Creating web server using microcontroller.   
7. Reading & responding web client data requests.  
8. Accessing and synchronizing RTC and NTP time servers.  
9. Lithium-Ion battery charging method

# Appendix

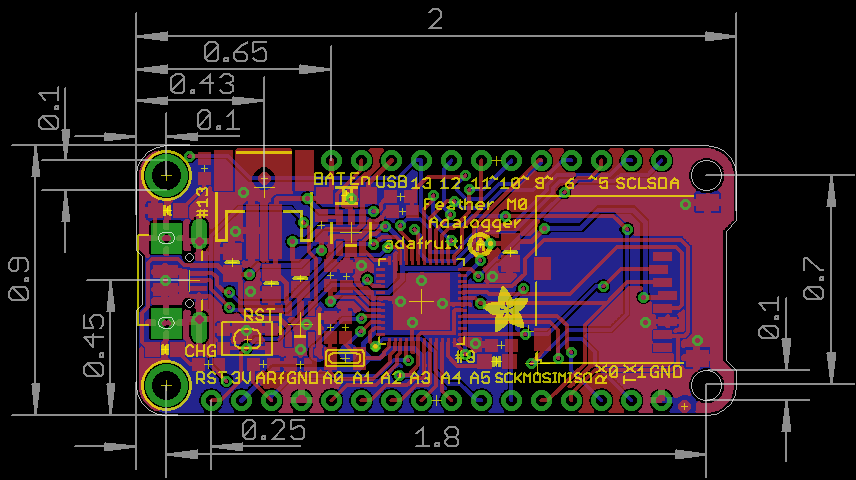
**Adafruit Feather M0 Board Pin Layout**



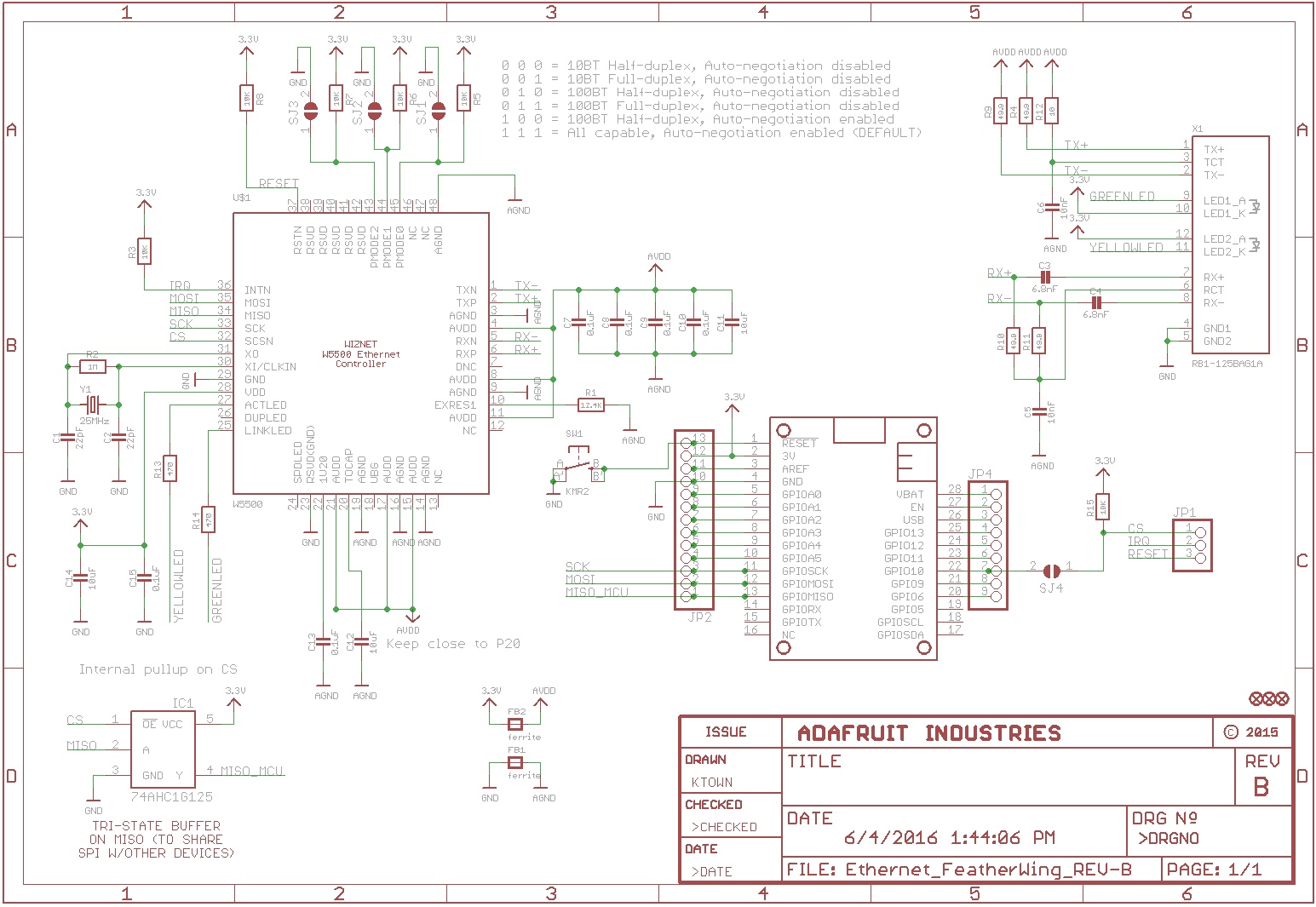
**Adafruit Feather M0 Board Schematic**

****

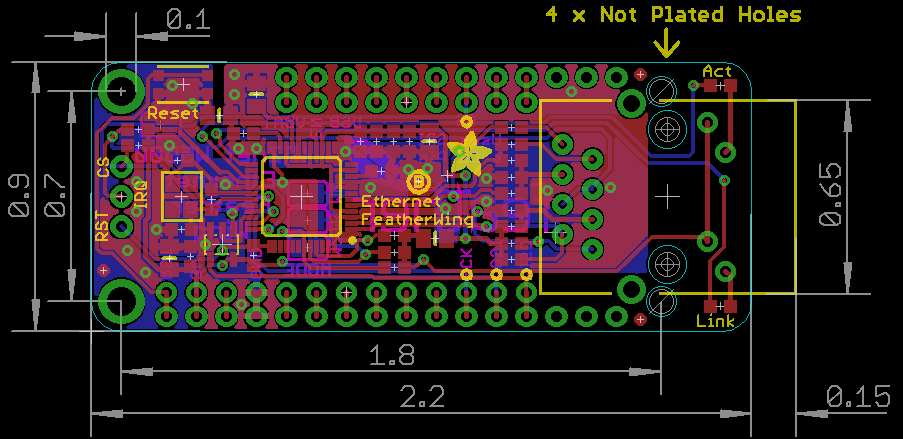
**Adafruit Feather M0 Board Layout**

****

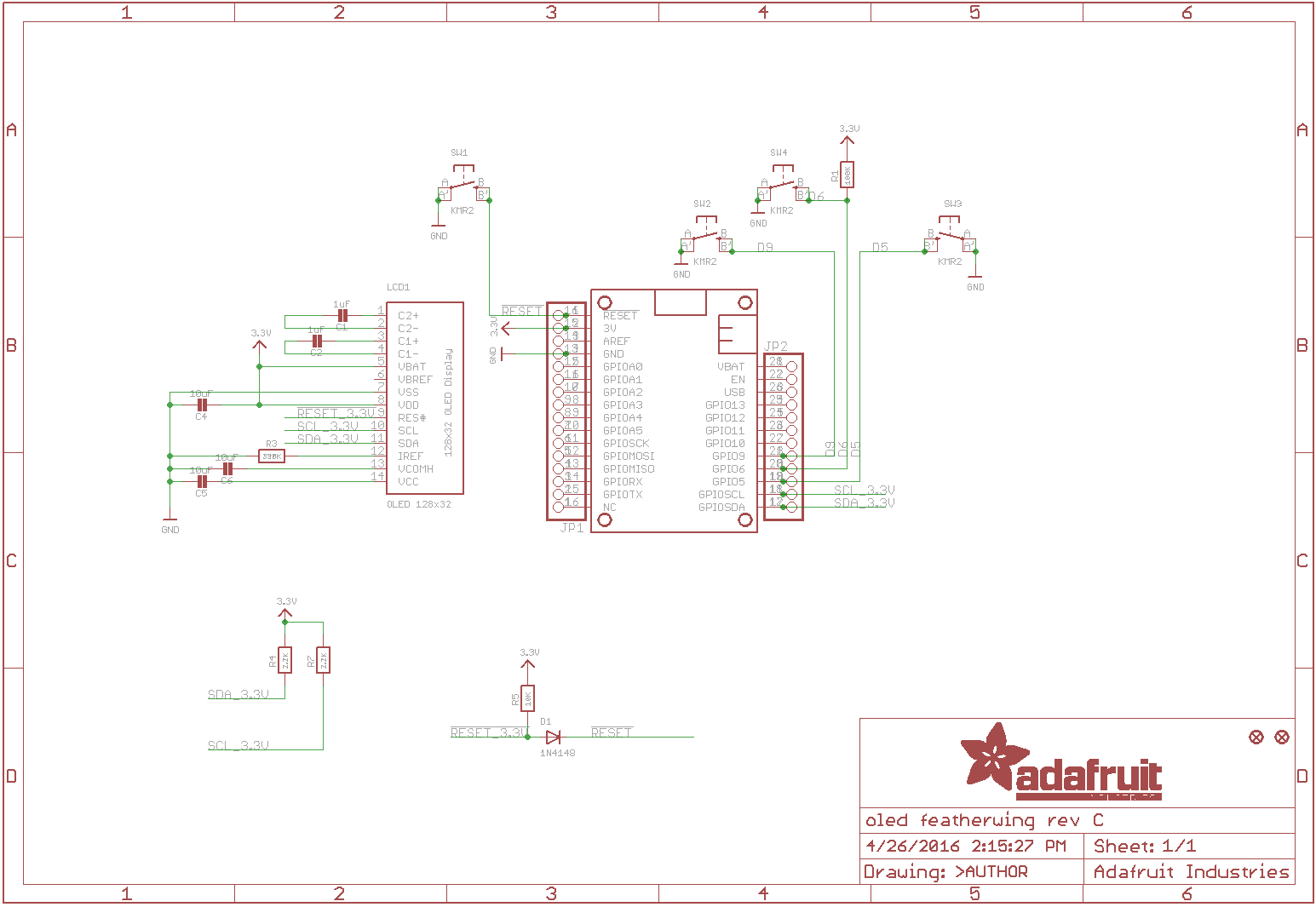
**Adafruit EthernetWing Board Schematic**

****

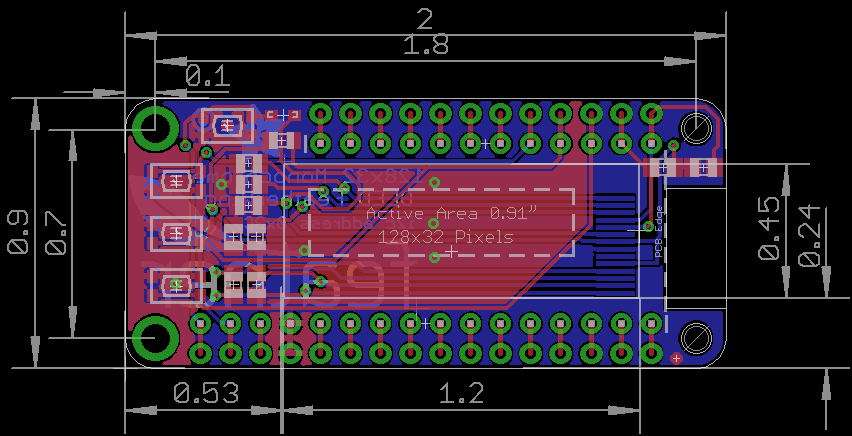
**Adafruit EthernetWing Board Layout**

****

**Adafruit Feather OLED Board Schematic**

****

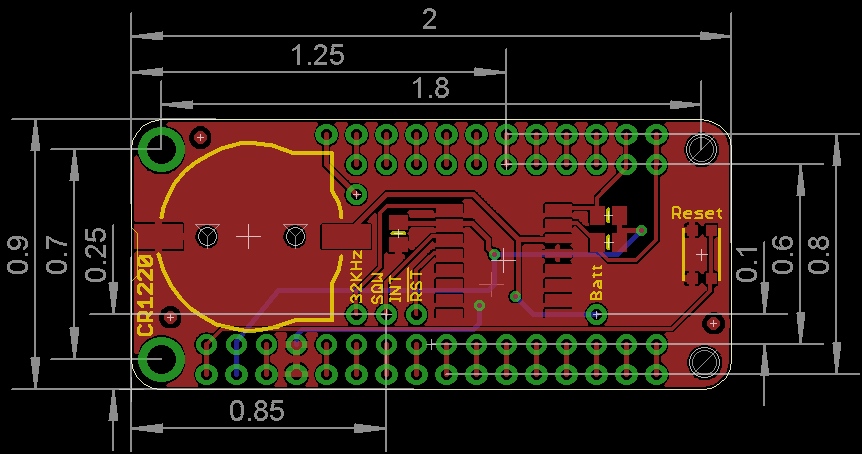
**Adafruit OLED Board Layout**

****

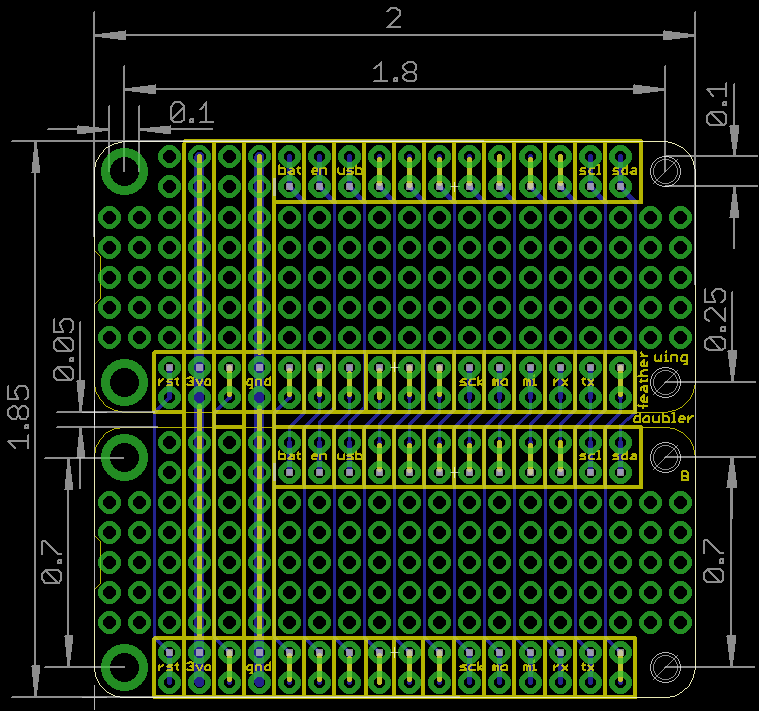
**Adafruit RTC Board Schematic**

****

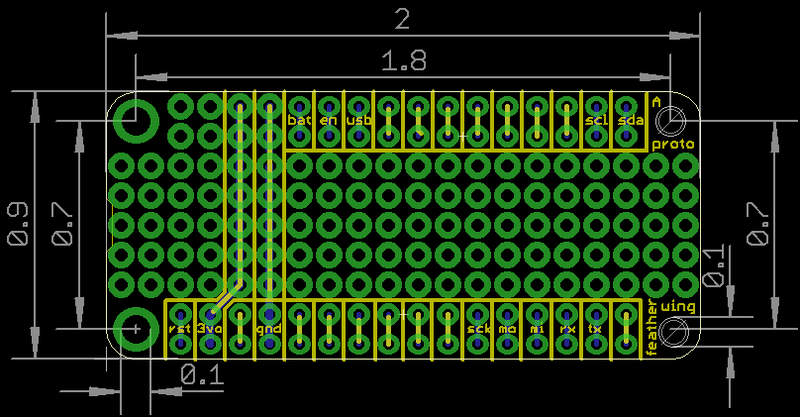
**Adafruit RTC Board Layout**

****

**Adafruit Feather Doubler Board Layout**

****

**Adafruit Protoboard Layout (Blank)**



**Adafruit Protoboard – Schematic: Modified for USB voltage monitoring and panel push buttons**



**Source Code - Attachment**

1. **Main Program Code**
2. **MsgModbus library**
3. **MsgModbus header file**