MAWS Software User Guide

Cameron Richardson

cr81@hawaii.edu

University of Hawai’i at Mānoa

Department of Oceanography

Introduction

This document serves as a user manual for the operation of the code hosted in the MAWS GitHub space. This user manual should only be consulted once the electronics of the MAWS have been completely assembled. The user manual is a living document and will be constantly re-visited as other functionality is added to the autosampler. Throughout grad school, I have spent my fair share (maybe even too much) of time looking through poorly constructed documentation and code. This user manual and the supplemental build guide for the autosampler were made in the hopes that future investigators spend less time wondering how the 'thing' works, and more time doing science with the 'thing'. Please feel free to reach out to me if you have questions, concerns or need help troubleshooting the autosampler. Note that while I am developed the code for the autosampler and lead the development of the autosampler itself, this project came to fruition through the work of many other people.

Acknowledgements

Camille Pagniello, Kaja Reinhardt, Maliheh Gholizadehsarvandi, Darin Hiraldo, Leilani Phan, Olivia DeCroix

Version History

V1 – Initial documentation (2026-22-01)

Table of Contents

Introduction *1*

Acknowledgements **1**

Version History **1**

‘conductivity.ino’ **3**

Setup 3

Calibration 3

Parameter Configuration 4

‘set_clock.ino’ **5**

Setup 5

‘control.ino’ **6**

Setup 6

Operation 6

‘conductivity.ino’

Setup

1. Ensure the USB-C cable is connected to the Arduino and your computer.
2. Open the ‘conductivity.ino’ script.
3. Ensure ‘rx’ and ‘tx’ pins are properly defined by checking their connections from the Atlas Scientific Isolated Carrier Board to the Arduino.
    a. e.g., if a jumper cable is connected to the ‘rx’ pin on the Atlas Scientific Isolated Carrier Board, you must wire it to the Arduino and define it as the ‘tx’ pin in the code.
4. On the top left of the Ardunio IDE program click the dropdown ‘Select Board’ menu and choose the ‘Arduino UNO R4’.
5. Upload the code by selecting the check mark next to the ‘Select Board’ menu.

Calibration

6. Open the ‘Serial Monitor’ in the ‘Tools’ tab of Arduino IDE.
7. Ensure your baud rate is set at ‘9600’ and ‘Carriage Return’ is selected.
8. You should see the following output (with unique numbers following each parameter):
    a. ‘-> EC: 78987’
    ‘-> TDS: 124’
    ‘-> SAL: 35’
    ‘-> GRAV: 5’
9. In the serial monitor, type:
    ‘C,1’
10. Ensure the Atlas Scientific EC probe is *completely dry.*
    a. Wait for the ‘EC’ reading to stabilize (5 min)
    b. In the serial monitor, type:
    ‘Cal,dry’
    c. You should receive the following response:
    ‘-> *OK’
11. Completely submerge the EC probe in the ‘Atlas Scientific 12,880 Conductivity Solution’.
    a. Wait for the ‘EC’ reading to stabilize (5 min)
    b. In the serial monitor, type:
    ‘Cal,low,12880’
    c. You should receive the following response:
    ‘-> *OK’
12. Thoroughly rinse the EC probe in DI water.

13. Completely submerge the EC probe in the ‘Atlas Scientific 80,000 Conductivity Solution’.
    a. Wait for the ‘EC’ reading to stabilize (5 min).
    b. In the serial monitor, type:
       ‘Cal,high,80,000’
    c. You should receive the following response:
       ‘-> *OK’

Parameter Configuration

14. In the serial monitor, type:
    ‘O,?’
    a. Based on what parameters the probe is logging, you will receive a different response
        i. e.g., if all parameters are enabled you will receive the following response:
           ‘-> ?,O,EC,TDS,S,SG’
    b. We are only interested in logging salinity (S), so based on your initial response, remove the other parameters.
        i. e.g., using the same example as above, type (enter each command separately into the serial monitor):
           ‘O,EC,0’
           ‘O,TDS,0’
           ‘O,SG,0’
        ii. Each time you remove a parameter you should receive confirmation:
           ‘-> *OK’
    c. In the serial monitor, type again:
       ‘O,?’
    d. Now all we should be left with is the output:
       ‘-> EC:35.00’
       ‘-> TDS:’
       ‘-> SAL:’
       ‘-> GRAV:’
       ‘->’

‘set_clock.ino’

When you initially set up the MAWS, the real-time clock (RTC) will have not been initialized and, once receiving power, will default to ‘2000-01-01-00:00:00’, which is not of much use to us unless you’re interested in a time travelling autosampler. As such I have written a small script to initialize the clock.

Setup

1. Ensure the USB-C cable is connected to the Arduino and your computer.
2. Ensure the CR1220 RTC battery is fixed in place on the Arduino.
3. Open the ‘set_clock.ino’ script.
4. On the top left of the Ardunio IDE program click the dropdown ‘Select Board’ menu and choose the ‘Arduino UNO R4’.
5. Upload the code by selecting the check mark next to the ‘Select Board’ menu.
6. Open the ‘Serial Monitor’ in the ‘Tools’ tab of Arduino IDE.
7. Ensure the baud rate is set to ‘57600’ and ‘Carriage Return’ is selected.
8. You should see an output in the serial monitor that resembles something like this: ‘-> Date: 2000-1-1 DOW: 1 Time: 00:00:00’
9. Use an external clock that is showing the current time and set the correct time using the format ‘YYMMDDwHHMMSSx’ on the clock.
    a. e.g., For setting the time to: ‘2025-10-03 Friday 16:30:50’, type the following in the serial monitor:
    ‘2510035163050x’
    b. You should receive the following response:
    ‘-> RTC time updated!’

‘control.ino’

Once the clock has been set, and the conductivity sensor has been configured, we can proceed with the operation of the main file, ‘control.ino’. I have tried to comment it very clearly, and the logic is quite simple. I initialize all of the required sensors first, followed by the pump logic, then the data storage functionality, and then the web page stuff.

Setup

Consult lines 13-18. There are very few things in the code that need to be tweaked, and lines 13-18 should guide you to those places. There are only a few nuances about the code that you should be aware of when operating the autosampler.

1. You must be less than 3 feet away from the autosampler electronics when connecting to its’ WiFi network.
2. When scheduling a pump to fire, consult the display RTC time at the top of the page, not your computers’ internal clock, as the autosampler schedules pumps to fire based on the clock it has imbedded.
3. Ensure that you schedule pumps to fire in the ‘future’. I usually schedule pumps to fire with a ~1 minute buffer. The pumps will not fire if the scheduled time is before the current time.
    a. e.g., Current RTC time: 01/01/26 – 10:00:00, if pump 1 is scheduled to fire at RTC time: 01/01/26 – 9:58:00, it will not fire.

Operation

4. Ensure you have read the setup nuances thoroughly.
5. Ensure the SD card is fixed into the SD card slot on the Arduino.
6. Ensure the USB-C cable is connected to the Arduino and your computer.
7. Ensure both of the interrupt switches are enabling a full 24V of power to the circuit (check using a multimeter on the terminal block).
8. Upload the code by selecting the check mark next to the ‘Select Board’ menu.
10. Open the ‘Serial Monitor’ in the ‘Tools’ tab of Arduino IDE. Ensure the baud rate is set to ‘9600’ and ‘Carriage Return’ is selected. If you receive a weird output e.g., ‘?’, re-upload the code.
9. You should receive the following output:
    ‘->Wi-Fi AP ready’
    ->AP IP address: 192.168.4.1’
    ‘-> Pump Controller Ready!’
10. You can now connect to the autosampler using your device as you would with any other WiFi network.
11. Once connected to the network, enter the AP IP address that was printed in the serial monitor, into a browser window.