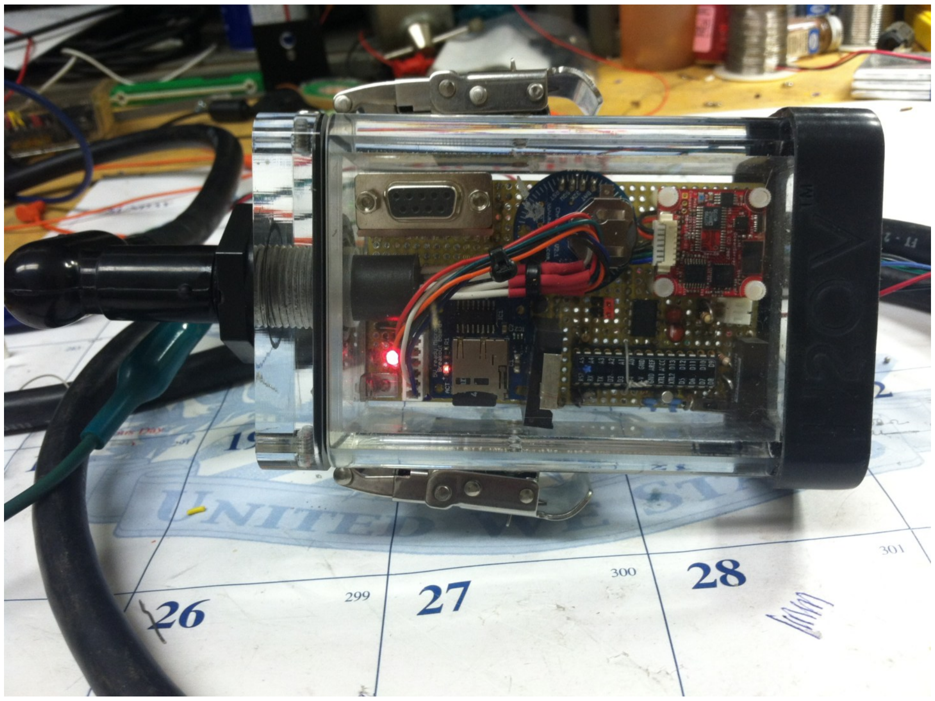
Arduino Based OS 5000 Datalogger



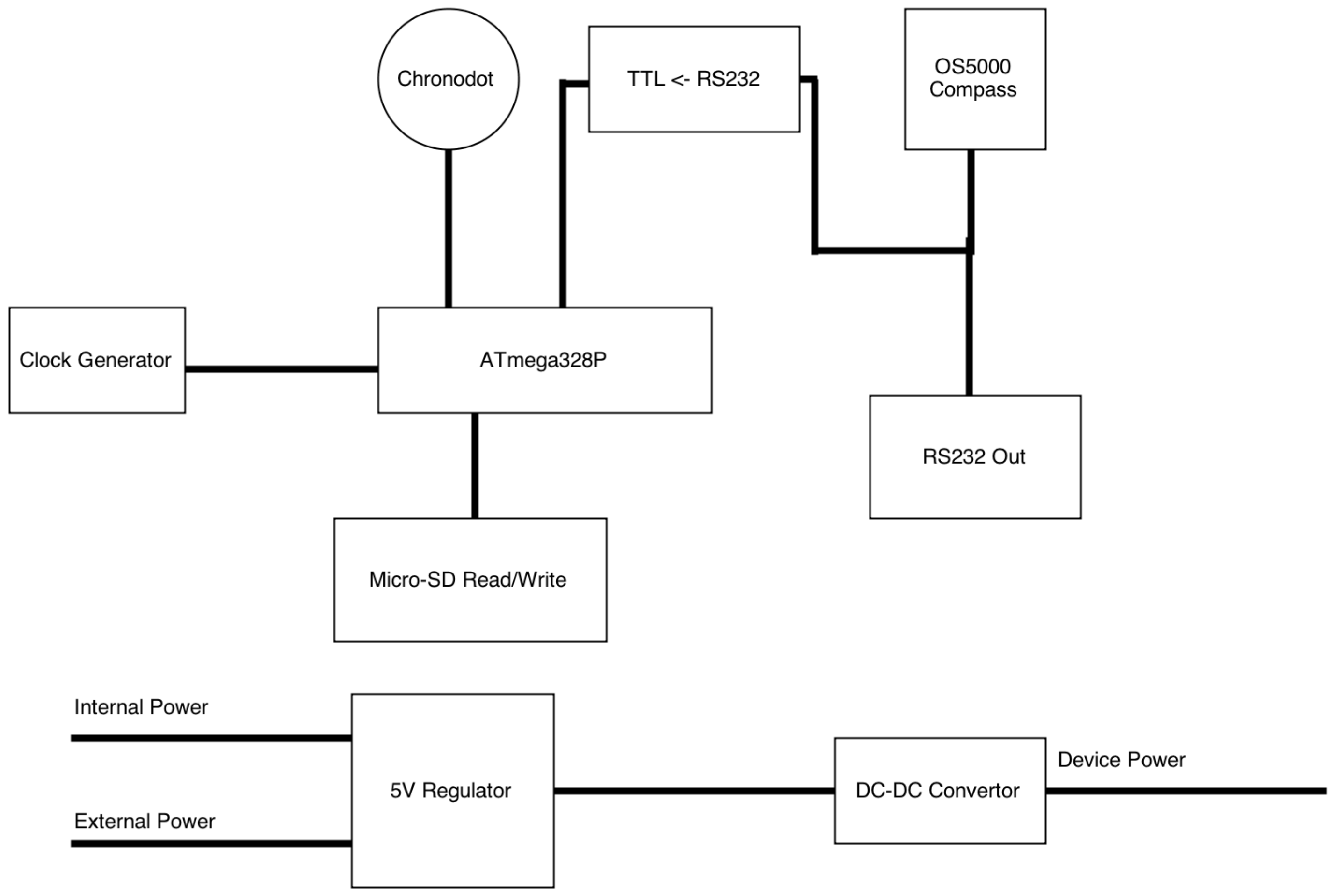
The compass, in its enclosure being externally powered, logging data.

This compass datalogger is designed to log timestamped heading, pitch, and roll data at pre-determined intervals to a micro-SD card. The main system relies on the OS5000-US compass for heading, pitch, and roll data. Compass data is level shifted from RS232 to TTL by the Sparkfun RS232 Shifter SMD. The date and time are taken from the onboard RTC - a Chronodot. Data is written to the micro-SD card via the Adafruit Micro-SD Breakout Board. Handling scheduling and communications between devices is the ATmega328P, installed with the Arduino bootloader for ease of reprogrammability. Logging frequency is determined by the RC constant of a 555 asynchronous timer. There are several options for powering the system, internal and external. With the included cable, the compass data may be read in real time, and settings and calibration by be adjusted in the field. With the board removed from the enclosure, a null modem cable may be used to communicate directly to the compass through the included DB9 connector.

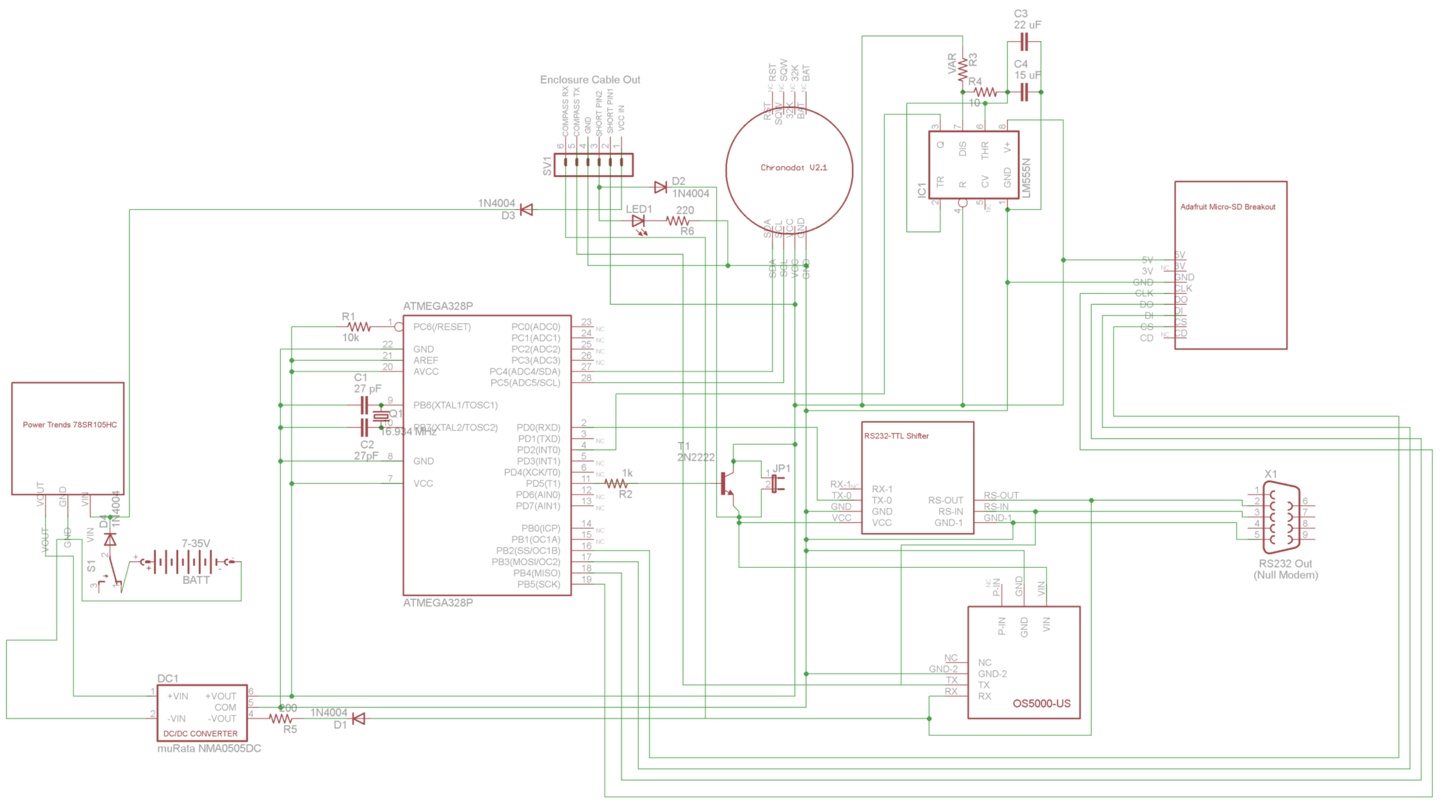
Power is managed on the board by both a DC—DC converter for those devices that require -5V, and the ATmega328P, which, when not logging data, turns off most of the peripheral devices and puts itself in idle/sleep mode. Current draw when idle is around 50mA, and when logging is generally around 130 mA.

Data entries written to the SD card are separated by commas, and follow the format of “Date Time, Heading, Pitch, Roll”. Sentence output is of the form “DD/MM/YY HH/MM/SS, Hxxx.x, Pxxx.x, Rxxx.x”. Three samples are taken in quick succession every time the logging period occurs, to prevent error. This can be changed within the provided code.

System Diagram



Schematic



BOM

1x ATmega328P

1x OS5000-US Compass

1x Chronodot V2.1

1x Adafruit Micro-SD Breakout

1x LM555N

1x Sparkfun RS232—TTL Board

1x muRata NMA0505DC (5v—5v DC—DC)

1x Power Trends 78SR105HC

1x 16 MHz Crystal

2x 27 pF

1x 2N2222

4x 1N4002

1x 37 uF

1x LED

1x 220 Ohm

1x 200 Ohm

1x 10k Ohm

1x 1k Ohm

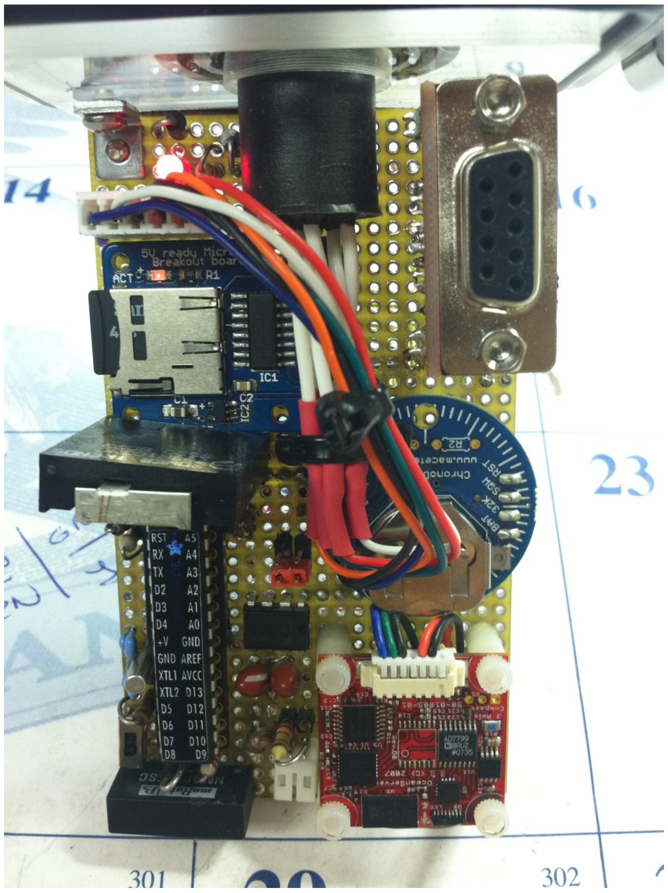
1x 10 Ohm

1x Timing Resistor

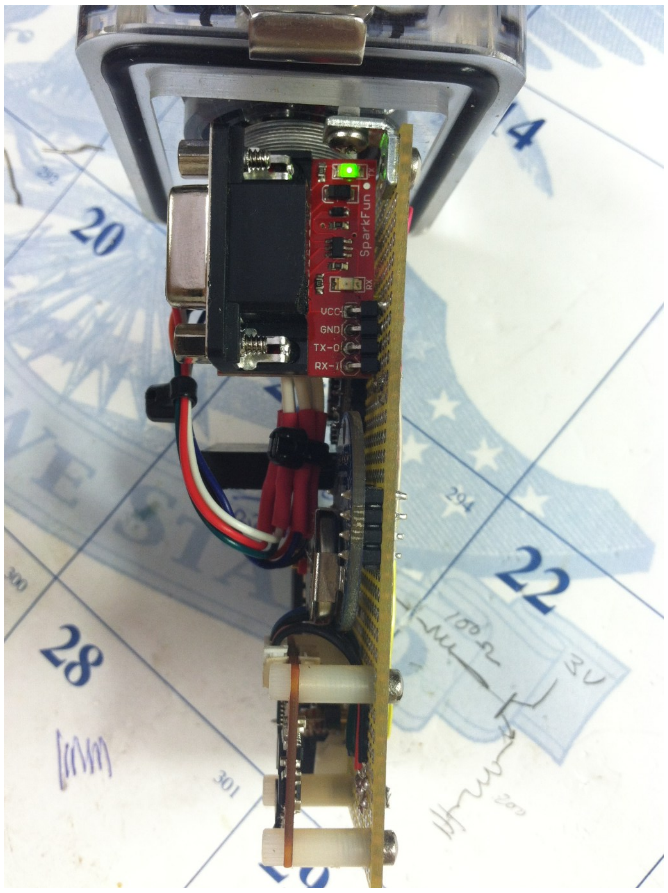
1x SPDT

Usage

Using the datalogger is relatively straightforward. To turn it on, simply flip the switch S1 near the bottom of the board to the on position (the position facing closest to the top of the enclosure), and the device should start logging data when powered by an internal power source. A green light on the RS232—TTL shifter means the compass is outputting data, and a red light on the SD card writer indicates that something is being written to the card. As soon as it is turned on, there should be three brief LED flashes on both the SD card board and the level shifter. This will occur whenever data is being logged.



Above: Top Right—Cable Status LED, with SD card read/ write LED below. Bottom Left—Power switch for internal battery. Center— Jumper JP1 in its "off" position.



Above: Top—Green Compass Data LED, indicating successful compass operation.

To set the logging period, an external resistor R3 is used to create an RC constant for the included timer. The resistor is socketed, and thus may be easily replaced by the user. The time between logging cycles is roughly determined by the given equation, where R is the given resistor.

$$t = .69*(R + 20)*.037$$

The clock serves to signal an interrupt on the microcontroller in order to wake it up from sleep. Upon waking, the microcontroller switches on the components that it turned off before it last went to sleep.

Before writing data to an SD card, it must first be formatted. This must be done prior to logging, on a computer. SD cards must be formatted with the FAT32 filesystem. To prevent any errors, this is best done with the official formatting tool provided by the SD Association, available from their website here: https://www.sdcard.org/downloads/formatter_4/ . Filenames written to the SD card will be in the form of YYYYMMDD.xxx, where YYYY is the year, MM is the month, DD is the day, and xxx is a numeric suffix, to prevent duplicates. Every time the logger is turned on, it will open a new file. Data is appended to the most recent file, until the current file exceeds 1 MB in size. When this occurs, a new file will be created whose numeric suffix is increased. Data will continue to be appended as normal to this new file. To retrieve data, remove the SD card from the board and insert it into a computer. Some example output looks like the following.

```
8/12/13 11:43:11, H179.3, P-5.4, R-0.6

8/12/13 11:43:12, H179.3, P-5.4, R-0.6

8/12/13 11:43:12, H179.3, P-5.4, R-0.6

8/12/13 11:53:9, H180.7, P-5.2, R-0.6

8/12/13 11:53:10, H180.8, P-5.2, R-0.6

8/12/13 11:53:10, H180.8, P-5.2, R-0.6

8/12/13 11:59:52, H180.9, P-5.4, R-0.6

8/12/13 11:59:52, H180.9, P-5.4, R-0.6

8/12/13 11:59:52, H181.0, P-5.4, R-0.6
```

The board is wired such that it is simple to read sentence data from and write to the compass in the field for calibration purposes. Make sure the six pin connector in the board is plugged in correctly—the blue mark on the side should match up with the one on the board. Only the provided cable should be used to connect to the device. Once the cable is plugged in, a red LED, LED1, should stay lit to indicate a successful connection. Plugging in the cable shorts the collector and emitter of the transistor that normally shuts the compass off, thus allowing the user to speak freely to the compass while the cable is plugged in. (This can be verified by a blinking green light when the cable is attached.)

To actually read compass data, plug the DB9 connector at the end of the cable in the RS232 port of a computer. Upon opening a terminal on the chosen COM port, compass data should update on the screen in regular intervals. An example output sentence should look like the following.

```
$C167.9P-8.7R-64.9*52
```

Where $ is the character indicating a sentence beginning, C indicates heading data, P indicates pitch data, R indicates roll data, and *xx indicates a sentence ending.

To send data to the compass, such as hard and soft iron calibration data as well as mounting data (and much more), press the Esc key, and the enter the appropriate code. Calibration information will not be explained here; it, and all the compass's other features, can be read about in depth in the OS5000 manual, which can be found alongside this document as well as online here: http://www.ocean-server.com/download/OS5000_Compass_Manual.pdf . For calibration information, see the section titled "Calibrating the Compass". Additionally, after mounting the compass, be sure to change the set compass orientation setting—see "New Mounting Positioning…"While there are many useful parameters to adjust, do not change the sentence format of the compass. Doing so may render the microcontroller unable to parse the data. Of course, the chip can be reprogrammed to accept different sentence formats using the Arduino IDE (more on that later). This may be desirable, as the compass can also output temperature, magnetic field strength, and acceleration data.

It is also trivial to read and write to the compass without the provided cable, through the on-board DB9 connector, located near the top of the board. When using this method to access the compass, a null modem cable **must** be used to connect to a computer, **not** a typical RS232 cable, although it is plugged into the PC as normal. Additionally, the jumper JP1 on board must be moved between the two pins closest to the top of the board, in order to override the switch that regularly keeps the compass off when not in use. When finished reading the compass in this way, make sure to put the jumper back into its original position, so that the device may function normally.

The ATmega328P has the Arduino bootloader installed, allowing the chip to easily be reprogrammed . To take advantage of this, a method of programming the chip from a computer is needed, such a USB enabled ISP programmer, or more easily, a compatible Arduino device. To program the chip, the Arduino IDE must be installed, which is available here: http://arduino.cc/en/main/software . Once done, the included code for the device can be loaded, edited, and uploaded to the chip, which should be seated in the Arduino. (The Arduino Uno is the cheapest and most common variant.) After reseating the chip back into the datalogger, make sure that all pins of the ATmega are making contact with the socket. For more information on the Arduino and how to program it, there are plenty of guides online. Here is a useful reference manual: http://arduino.cc/en/Reference/HomePage .

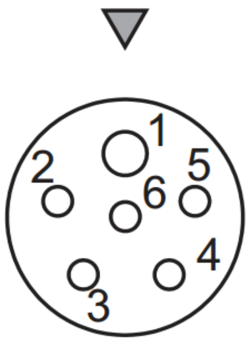
If the time and date ever need to be adjusted, (for DST or if it reinitializes), it can be reset through the ATmega. To do this, unseat the chip and place it in an AVR programmer (Arduino or ISP). The included code already contains subroutines to set the time and date, but they must be called in order to run. To do this, first find the routine "void loop()", and uncomment the "set_date()" and "set_time()" function calls. Then, look near the top of the code where the time variables are initialized. The variables "seconds", "minutes", "hours", "date", "month", and "year" should be initialized to the time to which the RTC will be set. Do not use leading zeroes when initializing the variables. "day" is the number of the day in the week, and need not be filled in, as it is not used. To accurately set the time of the RTC, program the time to be a few minutes into the future. After compiling and writing this code to the ATmega and reseating it in the datalogger, wait until the time specified earlier, and then turn on the device, as the time will be set immediately upon the device turning on. This ensures that the time that is set is indeed the current time. After doing this and making sure the device has logged at least one sample to the SD card, turn off the device, and check the SD card to see if the time and date logged in the most recent file match up with the time you set. Remove the micro from the board, and comment out the "set_date()" and "set_time()" calls once again. Reprogram the microcontroller with this code, and it should function normally again, using the correct date and time based off of the given parameters.

The Arduino also makes it easy to change how many samples of data are logged every logging cycle. To change this from the default of three, simply open up the Arduino sketch associated with the device, browse to void_loop(), and change the variable "times" from its current value to however many samples should be logged every cycle.

The board can be powered in two ways, by a source inside of the enclosure, or an external one. There is diode protection for both sources to prevent failure. Power from any source is routed through a 5v regulator, the acceptable input range of which is 7—30 V DC. From there, a DC-DC convertor is used to pull –5v for those devices that need it. To use an internal power source, simply plug a battery into the female JST connector near the bottom of the board. To turn it on when powered in this manner, simply flip the switch on board into the on position.

To provide power from an external source, power must be run into the enclosure through a cable. The included cable provides a means to do this, however this cable is not a viable option in the field, as it is wired to be used for communicating with the compass via PC for calibration purposes, thus when it is plugged into the device, the compass will not turn off. An appropriate cable can be made based on the cable pinout explained later in the documentation.

When looking straight ahead at the male pins on the bulkhead, the pinout is as below, with the largest pin on top. For reference, the type of cable used here is the RMG-6-BCL, with more data available here: http://seaconworldwide.com/wp-content/uploads/2011/07/RM_RMG-FS_and_Associated_Parts.pdf .



Device Pinout

Pin 1—Transmit (Data from Compass)

Pin 2— Shorting Pin

Pin 3—Receive (Data to Compass)

Pin 4—Shorting Pin

Pin 5—Ground

Pin 6— Power In

An idea for powering the board internally with a battery is to use two LiPos in series to generate a sufficient voltage. This could be accomplished easily using two of https://www.sparkfun.com/products/8483 . They can be interfaced with one another through JST connectors, available in both male and female varieties here: https://www.sparkfun.com/products/8670 , and here: https://www.sparkfun.com/products/8613 respectively.

For more details on included peripheral hardware, (especially the compass and its calibration functions), please refer to the official manuals/documentation provided with this file.

Individual Device Information:

Arduino Uno: http://arduino.cc/en/Main/arduinoBoardUno

OS5000-US: http://www.oceanserver-store.com/osevkitsorus.html

Chronodot: http://www.adafruit.com/products/255

Adafruit Micro-SD Breakout Board: http://www.adafruit.com/products/254

Sparkfun TTL—RS232 Shifter: https://www.sparkfun.com/products/449