Practical Arduino Data Loggers

By The Curious Questioner

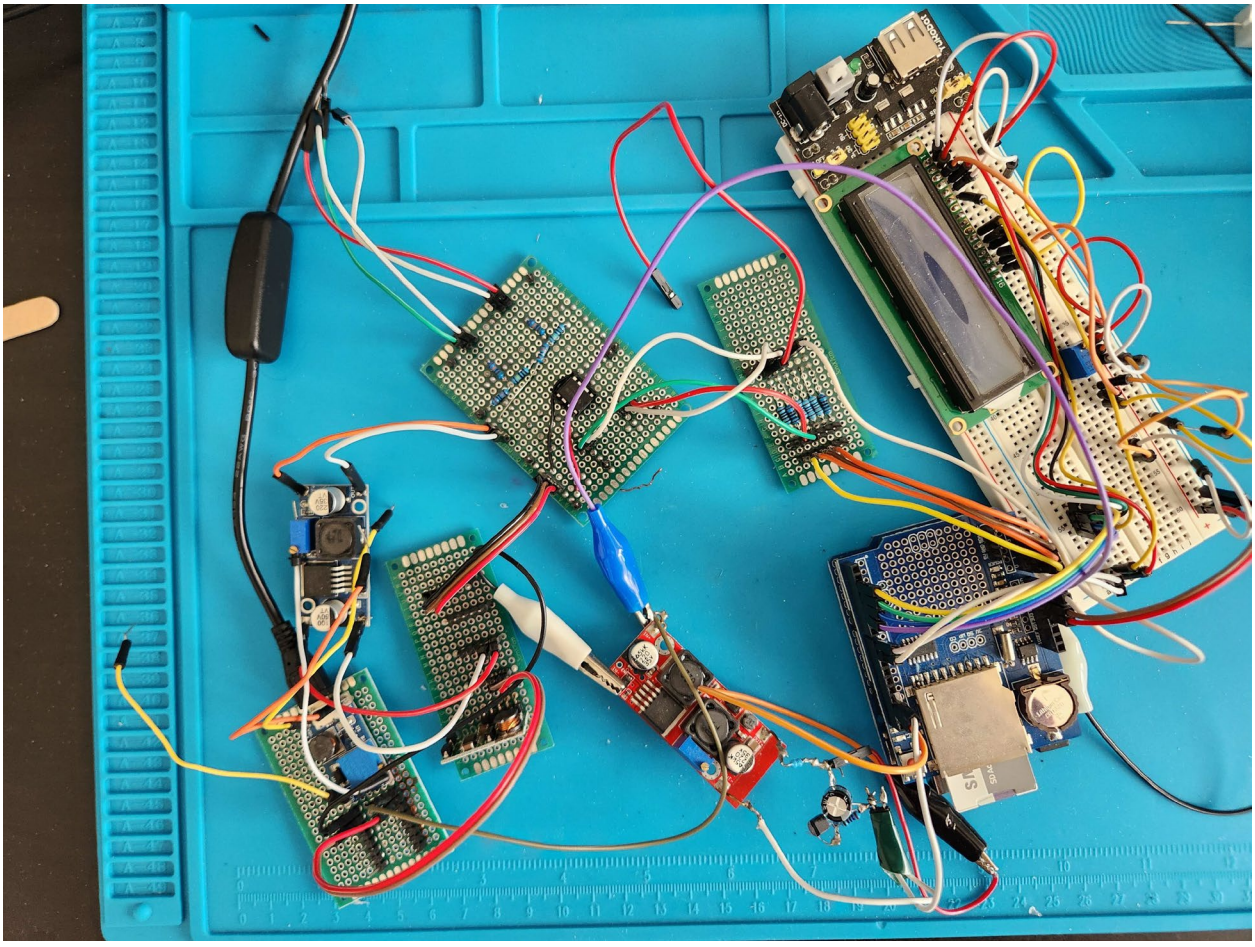

Table of Contents

Scope

In this paper, I describe how I came up with an Arduino UNO data logger that is practical, low cost, easy to make and usable in a variety of applications. I'm not a pro nor a full-time programmer – just someone with an interest in the field and happy to share my work with other interested parties. I will attempt to describe my work in a simple form assuming the reader has a basic understanding of Arduino and programming.

My code doesn't process real time data: all the data is recorded first, then saved on a micro flash card and then visualized (and processed) after the recording is complete.

In a future paper, I will share the python code used to process the recorded data (this code can also be used to read any .csv or .txt file).

Introduction

I got started with Arduino because I wanted to build a remote-controlled drone. But as I was learning more about the Arduino code, I realized it was much easier to learn than I had originally thought, so I put my big project on hold to investigate motor control projects. Being an electrical engineer, I wanted to experiment with some of the circuits that I have studied back in college.

As I was going through these first experiments, I realized the importance of having a way to record and visualize the data. I was using scopes and multimeters, but they had some limitations when it came to saving the data. This pushed me to look into making a functional data logger that would be fast enough to capture and record the data.

And so, I ventured into a new journey looking on the various forums for data logger codes and examples: I found a lot of information online via YouTube videos and various forums in bits and pieces that gave me partial answers, but not what I was really looking for. By selecting the information I found and putting it all together, I managed to write the code that would do what I originally had in mind.

I'm extremely grateful to all the authors of videos and forum explanations that I list in the "References" section at the end of the paper including the various participants of the Arduino.cc Forum community. A special thanks goes to Bill Greiman for sharing his awesome analog bin code that I used for one of the data logger projects. Although I never chatted with or met with any of these wonderful people, I learn everything I know from them.

How to Read this Paper

This paper is organized with a table of contents and titled paragraphs to aid the reader in digesting and jumping to topics of interest. The explanation of the code is included in the commented lines of the code file and not in the body of this document.

I focused on the details of my project and not the theory behind each component or concept of each topic. If you are new to Arduino programming and need to familiarize yourself with Arduino or with electronics, the sources that helped me to master each topic are listed in the "Reference" section at the end of the paper. These sources were of great help to me and hope they will be for you as well.

The Making of the Data Logger

In this section, I provided information about the finished product and list of components that I used. You can find each component on any online outlet: I got all my components from AliExpress.

I made two versions of the data logger: one without a display that you connect to your computer and one with an LCD display and power supply that works independently (without the need to connect to a computer). The latter can be powered either with a battery or with a plug-in AC power supply. For each version, the code can sample at 64 samples per second.

For the Analog Bin logger code is possible to sample at a much higher rate: I've sampled sine waves up to 1080Hz. For this code, I made only one version of the data logger that needs to connect to the computer.

Both versions are discussed in the following.

Functions and Limitations of the Data Logger

Based on my experiments, the following are a list of my observations on the functions and limitations of the Arduino UNO data logger that I made.

- There are limitations in recording the data when using more than 4 channels. These limitations are mainly two factors:
    1. The SD card writing time increases causing a delay in sampling and yielding less sampling points.
    2. A4 and A5 pins are respectively the I2C pins SDA and SCL. These pins are always high and can't be used as input pins (or at least I didn't find any information on how to reprogram these pins to sample data).
- The Arduino UNO analog pins can measure values ranging between 0 and 5 VDC. Sensor kits and devices made for Arduino UNO are made to output a signal within this range. Any other signal outside this range will require an additional stage before connecting to the

analog pins. For the purpose of my tests, I used a signal with a DC bias to shift the waves in the 0-5VDC range.

- The Arduino analog pins need to be decoupled, isolated from each other and from ground (with 10k resistors) to allow better measurements: this required an additional circuit board shown in Figure 1. I found out about this the hard way: while reading the recorded data, I noticed that when recording on one channel, the other channels were not at 0 but were also recording low fluctuating values.
- The Arduino's ADC default resolution is 10-bits, so the range is of 1024 points. (It's possible to change it to a maximum of 12 bits, as explained in the Reference section, under "Arduino Pins".)
- The Arduino default reference voltage (Vref) used to read measurements is very unstable. The internal Vref is better, but the best approach is to use an external Vref. The downside is that an additional external circuit is required.
- For all data loggers presented in this paper, the data is recorded directly in bits and the conversion to voltage values and any calculation of RMS values is done post processing. This is so precious sampling time is not wasted. The RMS calculations I did post processing agree with the voltmeter readings and within the values displayed in my scope.
- Samples are assumed to be taken at equal intervals. In reality, the SD card has delays due to its variable latency that take some samples at different intervals. There are also delays in the Arduino UNO data processing computation. I found that the error in the final measurements and RMS calculations are tolerable.
- I didn't record the RTC value for each sample because it takes 50ms extra to record each value: that drastically reduced precious sampling time. The time series is generated post processing via the python code assuming all samples are equally spaced in time.
- Data visualization during recording reduces sampling rate. For this reason, I display only the status of the recording and error message (either on serial monitor or LCD screen) before (or after) the sampling starts.
- To sample at very high rates, you need to learn how to program the registers of the Arduino UNO. This is beyond my level of understanding, so I didn't pursue it for now.

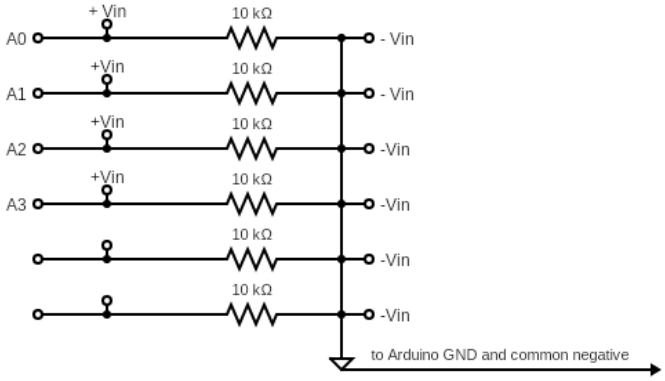

Figure 1 - Arduino UNO analog inputs isolation board

Data Logger Connected via USB to the Computer

This is the basic data logger and to make it work you need to connect the Arduino UNO to the computer via USB cable.

I used this version to develop data logger hardware and the Arduino UNO code. It allows you to monitor what is going on via the serial monitor and serial plotter and to verify quickly that each shield is working properly by running the example codes.

I prefer this version if I'm working with my laptop, and on a new project. It's easy to connect everything together and to check that each component is working properly. Also, I can turn data logging on and off from the computer.

The data logger is assembled with the following components:

1. Arduino UNO
2. Analog Pin input isolation board
3. Mini data logger shield with SD Card Reader and RTC
4. Input power board
5. Power distribution board
6. Vref-external power source (power + filter)
7. Amplifier power supply
8. Voltage Bias power supply
9. Amplifier input boards

The code for this board is available on GitHub.

Figure 2 shows the basic data logger that I assembled with the electronic boards to step-down the signal (as I will explain in the following). You can use this same arrangement without these input boards if the signal you need to record is within the Arduino UNO's 0-5VDC range.
Also, the mini data logger shield shown in Figure 2 is a circuit board with both a micro-SD card and an RTC module. This shield can be used in lieu of a separate SD card reader and a separate RTC module, giving the same results.

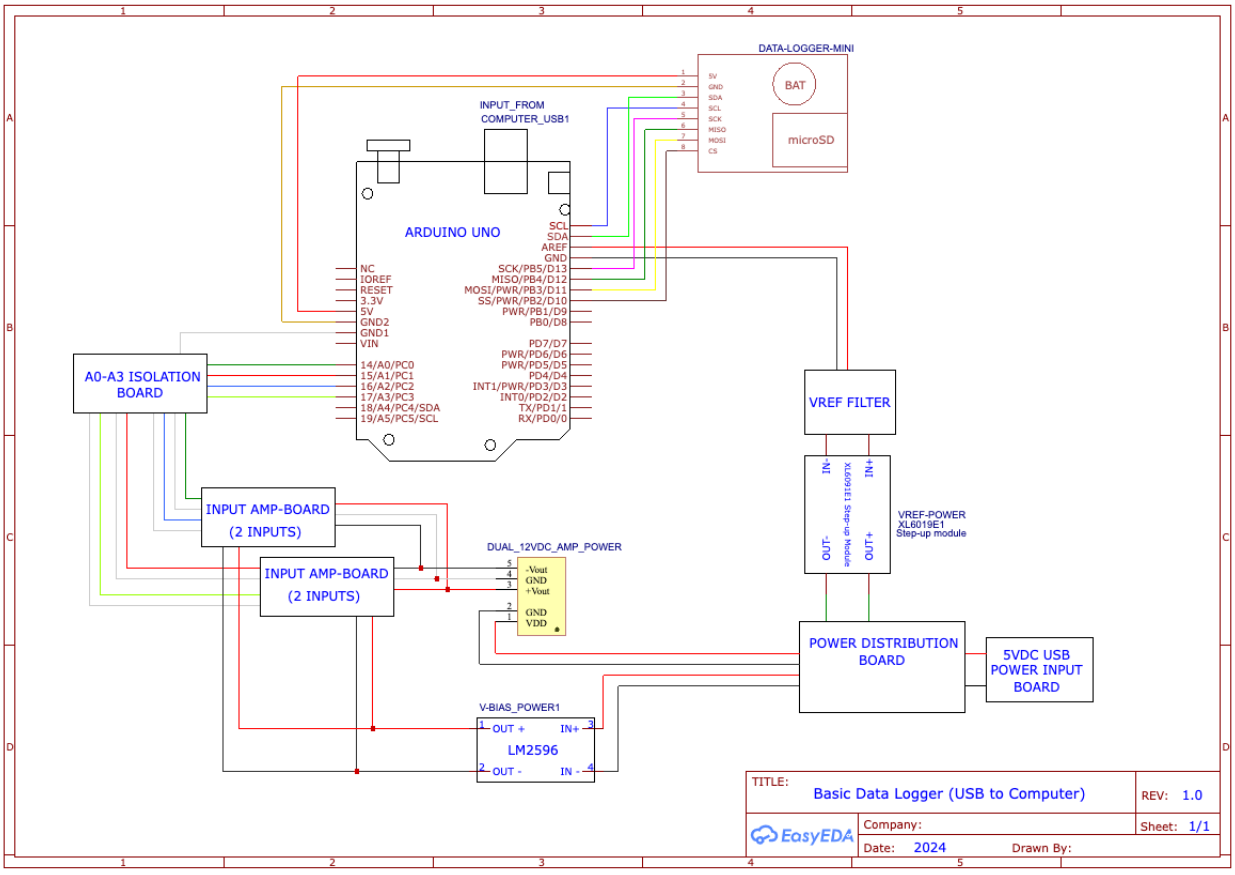

Figure 2 - Data Logger Connected via USB to the Computer

Data Logger with Modified Analog Bin Code (Connected via USB to the Computer)

This data logger uses similar components to the data logger shown in Figure 2 except with the additional control & status board. This additional board allows you to start & stop the recording and to monitor the status and overruns with the LEDs.

I use this type of data logger only with the Analog Bin Code and connect the Arduino UNO to the computer via USB. The code is written to output status messages on the serial printer. I've used this code successfully with the RTClib by Adafruit version 2.1.3 and the SdFat library – Adafruit Fork up to version 1.5.1. The code didn't work with the SdFat library of higher versions.

The modified Analog Bin code is available on GitHub. The original Bill Greiman's code is at this link.

With this code, the data is logged in a ".bin" file so in order to use it, it's necessary to convert it to .csv with the Python code available on GitHub.

Figure 3 shows this circuit configuration with the following components (the control board schematic is shown in Figure 4):

1. Arduino UNO
2. Analog Pin input isolation board
3. Mini data logger shield with SD Card Reader and RTC
4. Input power board
5. Power distribution board
6. Vref - external power source (power + filter)
7. Amplifier power supply
8. Voltage Bias power supply
9. Amplifier input boards
10. Control board with start & stop buttons and LEDs.

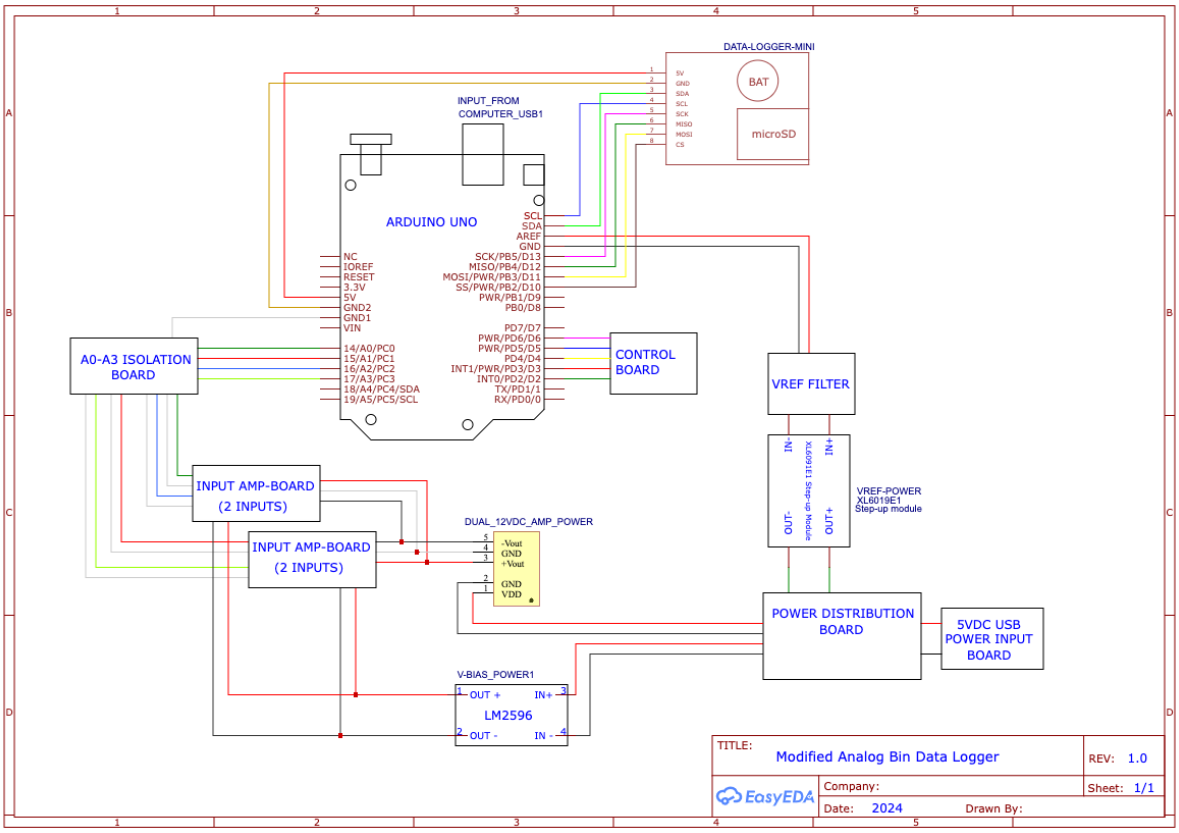

Figure 3 - Modified Analog Bin Data Logger

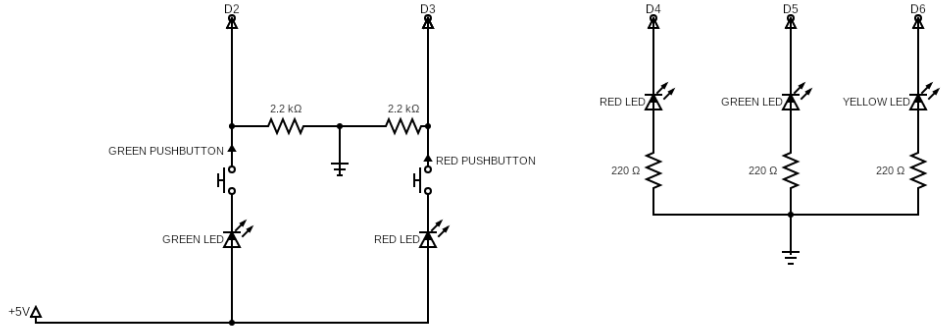

Figure 4 - Control Board Circuit for the Modified Analog Bin Data Logger

Data Logger with LCD Display and Power Supply

This is the data logger that I use when I need to log data remotely without using a computer. This comes in handy for outdoor projects and for logging data for a long time (such as 24hr or more).

This version of the logger doesn't require me to be physically on site and allows me to deploy the device practically anywhere. I will use it on my future drone project to monitor its flying performance and record other sensor data while in flight.

This version has its own power supply and an LCD diagnostic display that will show you if the data logger is functioning properly or if there is a problem.

The complete data logger is shown in Figure 5 and is composed of:

1. Arduino UNO
2. Analog Pin input isolation board
3. Mini data logger shield with SD Card Reader and RTC
4. Input power board
5. Power distribution board
6. Vref-external power source (power + filter)
7. Amplifier power supply
8. Voltage Bias power supply
9. Amplifier input boards
10. LCD Display.

[The code is available on GitHub.]

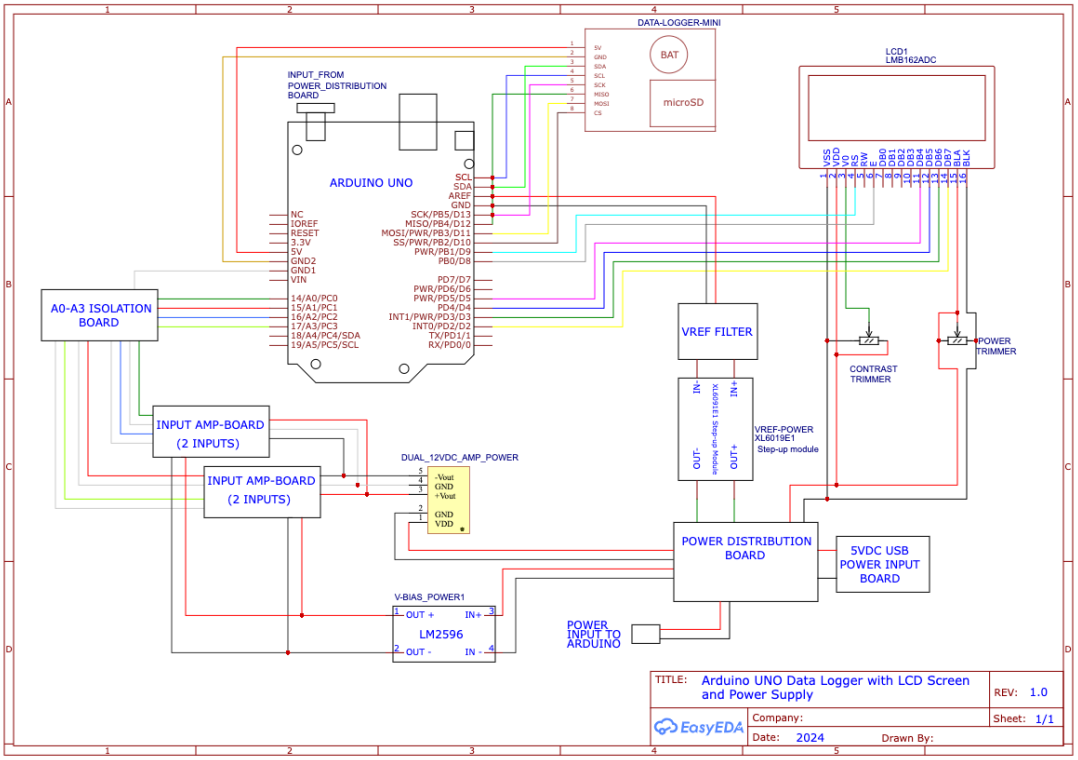

Figure 5 - Arduino UNO Data Logger with LCD Screen and Power Supply

Sampling Capabilities of the Data Logger

When I was looking for the code on how to record the signals from the analog pins, I didn't find a good resource that would clearly state the actual Arduino UNO sampling rate. I only found sources that would state the maximum (theoretical) sampling rate of the Arduino UNO board or of the other input voltage and current boards compatible with Arduino UNO. When I started running the code, I noticed that the actual sampling rate was much lower than the theoretical one. This prompted me to test each code's actual sampling capabilities.

I also discovered that to achieve a high theoretical sampling rate you need to program the registers: something that honestly, I can't do. So, although I didn't develop the code to program registers, I share my resources in the References section at the end of the paper.

I used the Arduino UNO timing commands ("millis") to measure the time required by each part of the code (see commented lines in the code). That gave the value of the sampling time, the time it took to write to the SD card and the time it took to get the real time from the RTC. However, this method can't measure the total time to run the entire void loop. For this reason, I decided to just record the data, visualize it, and then count the actual sample points on the graph.

For the data logger code, I made different type of recordings: on a single channel, on two channels, and on all four channels. I recorded the data of four different waves (simultaneously for 4 channel recordings): sine wave, ramp, square and positive ladder. I recorded these waves at different frequencies to find the highest frequency that could be recorded (I'm sharing the data recorded).

For the Analog Bin code, I made recordings using only sine waves at different frequencies to check which was the highest frequency to be recorded at a decent sampling rate. I need this code to record AC currents at 60Hz and its harmonics, so I didn't explore other wave shapes. The graph in the following shows the results.

In the next two paragraphs I will share my results for each code.

Sampling Rate with the Arduino UNO Code (Connected via USB or Standalone with LCD display)

Based on the tests that I performed, I found the sampling rate to be 64 samples per second (sps) - or 15.625 milli-sec between samples - when recording on 4 channels simultaneously.

I measured 4 types of waves: sine, ramp, square and positive ladder. I took measurements by increasing the frequency progressively at 1Hz increments and then sampling the data. I found the following results (see Figure 6 for the graph):

- Frequency upper limit is 3Hz, which guarantees you are sampling at 64sps. You can stretch it to 4Hz, but you start losing some information. At 5Hz you really start missing points. (I recorded data up to 10Hz)
- For waves of 5Hz or more, the sampling rate starts to progressively decrease: at 10Hz I measured approximately 55sps.

- Recording 1 channel vs 2 channels doesn't impact the sampling rate.
- Recording 4 channels vs 2 channels decreases the sampling rate by approximately 8sps.

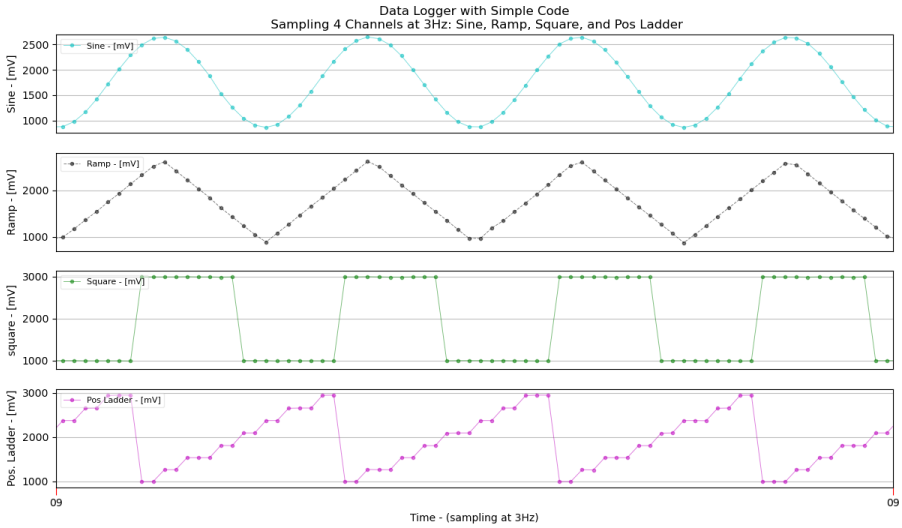

Figure 6 - Measured Signals at 3Hz - Approximately 64sps

These results showed me that I can use this data logger for typical projects using devices of experiment kits' sensors (such as temperature, sound, radar, moisture, and pressure sensors). It's also possible to use this Arduino UNO data logger to record voltages and currents of electrical transients in capacitive and inductive circuits, motor performance monitoring, and other events that last approximately less than 166ms. (In the reference section I provide resources that discuss sampling rate and the optimal sampling rate required.)

To log at a higher rate, you need to use the Analog Bin code that is explained next.

Sampling Rate with the Arduino UNO and the Modified Analog Bin Code

The Analog Bin data logger code was a real lifesaver!
This code allowed me to measure sine waves from 60Hz to 1080Hz (1080Hz is the 18th harmonic of an average sine wave).

Measured values are show in Figure 7 when setting the sampling rate at 10,000sps. During data recording, I noticed that there overrun errors once the sampling time goes over 30sec; these errors don't have a practical impact on the data recorded (I will investigate this in the future).
I will use this code primarily for my AC generator experiments. (I made a three-phase generator by coupling a set of hoverboard hub motors.

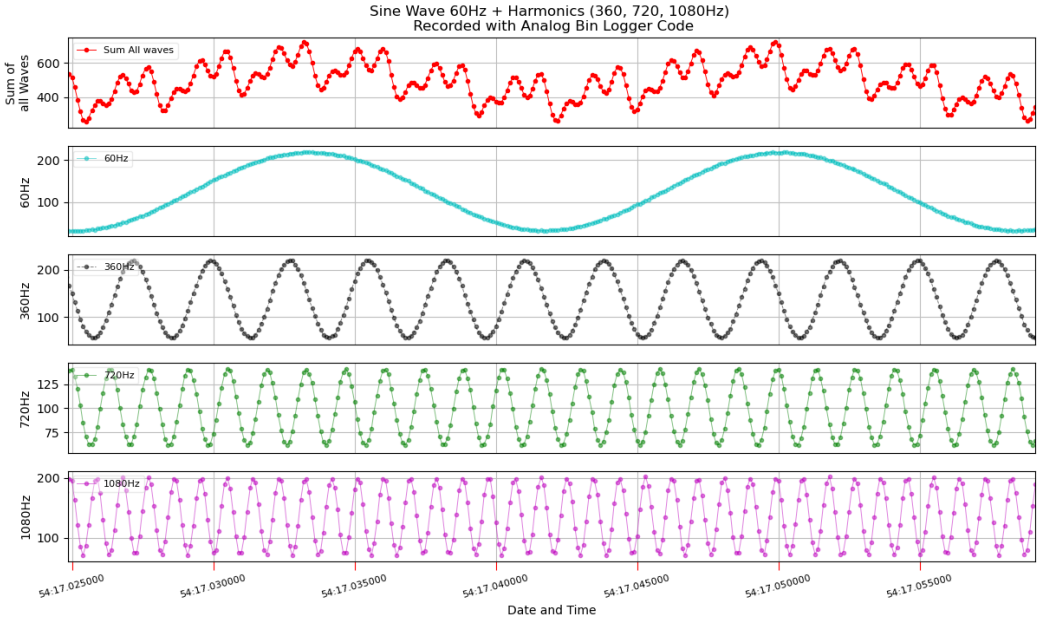

Figure 7 - Sine Waves Measured with Analog Bin Code at 10k SPS.

The Voltage Reference (Vref)

The Arduino analog pins take voltage measurements with respect to a reference voltage. The reference voltage can be the Arduino default reference, the internal reference, or an external reference voltage.

The source that gives more precise values is the external Vref, but to do this you need an additional power supply that gives you a steady voltage (see Andres Spiess' video in the references for a detail explanation on this topic).

The value of the reference voltage would depend on the source signal but in general, a lower value will allow you to measure with more precision. This is because the gain is Vref/1024 (in Volts/bit or mV/bit) and this value multiplied by the analog pin value (in bits) will give you the measured value. (All of this is well explained in the sources of "Vref - Arduino Voltage Reference" in the References section that I provide below.)

For this project, I chose an external reference voltage value of 4960 [mV] for some readings and 2064 [mV] for other readings. The resolution for each value is 4.850 [mV/bit] and 2.0176 [mV/bit] respectively. To obtain the Vref value, I used the TL431 fed by a power supply and provided with an output filter as shown in Figure 8. This circuit provides a steady value with a good precision. I also added an output capacitor filter to keep the voltage from fluctuating.

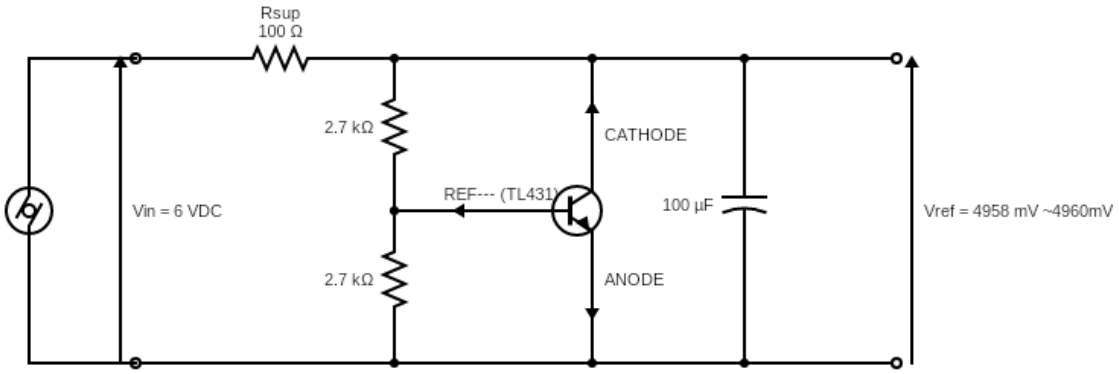

Figure 8 - Circuit for the Arduino UNO external voltage reference "Vref"

Depending on the project, I can regulate the Vref to a value that best suits my measurements. The Value of the Vref will depend on the signal to acquire since the Arduino has a range of 1024 bits. It's best to use all the bits for the entire range of the signal. (See videos in the reference sections of "Analog Pins and ADC Resolution" and "Vref - Arduino Voltage Reference" to know more.)

Electronic Boards to Amplify or Step Down the Signal

Since the Arduino can only measure voltage values between 0 VDC and 5 VDC at first it seemed impossible to make measurements of typical AC voltages and currents.

After researching on-line and refreshing my knowledge of electronics, I realized that I had to make two types of boards. One type to read DC values from zero to a higher voltage (outside of the Arduino UNO range) and another type to read voltages varying between positive and negative values (such as AC signals).

To read DC voltages varying from 0 VDC to a value greater than 5 VDC, I simply created a voltage divider board as shown in figure 9. The resistors are designed to fit the scaling required. (See the online Voltage Divider Calculator)

I bought the resistor, circuit board and other accessories on AliExpress. However, these can also be found on other outlets.

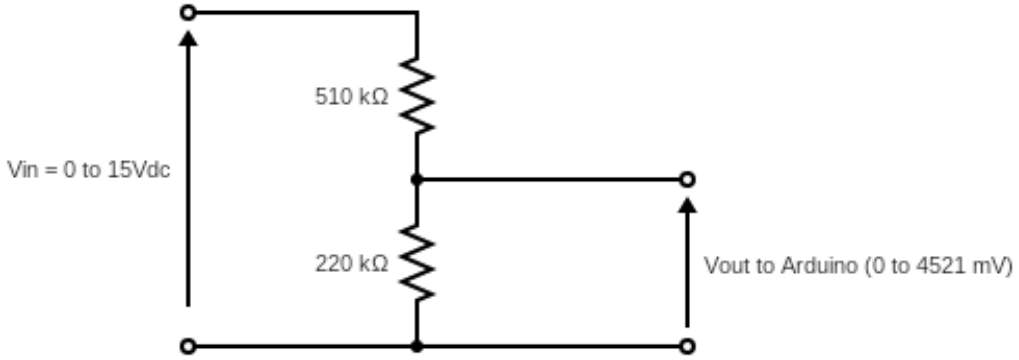

Figure 9 - Voltage divider circuit: step down a 0 to 15VDC input voltage to a 0 to 4521 mV voltage to Arduino UNO

To read DC positive and negative voltages or AC voltages, I scaled the original value and added a DC bias. I made a multi-stage board with an amplifier shown in Figure 10. (In the future, I plan to use this board to measure charge and discharge current cycles of a battery because they swing between positive and negative values). Figure 11 shows a similar circuit for a +/-10Vpp input.

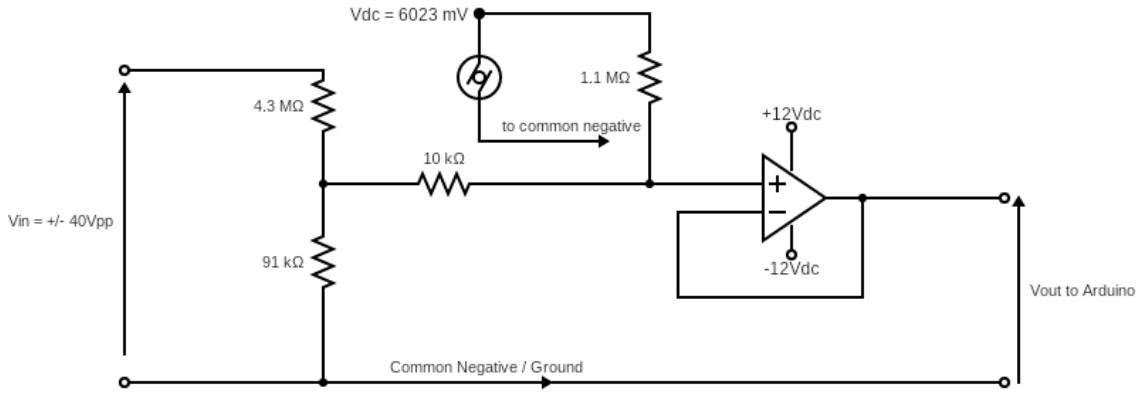

Figure 10 - Acquisition board for a +/- 40Vpp input signal with the TL072CP

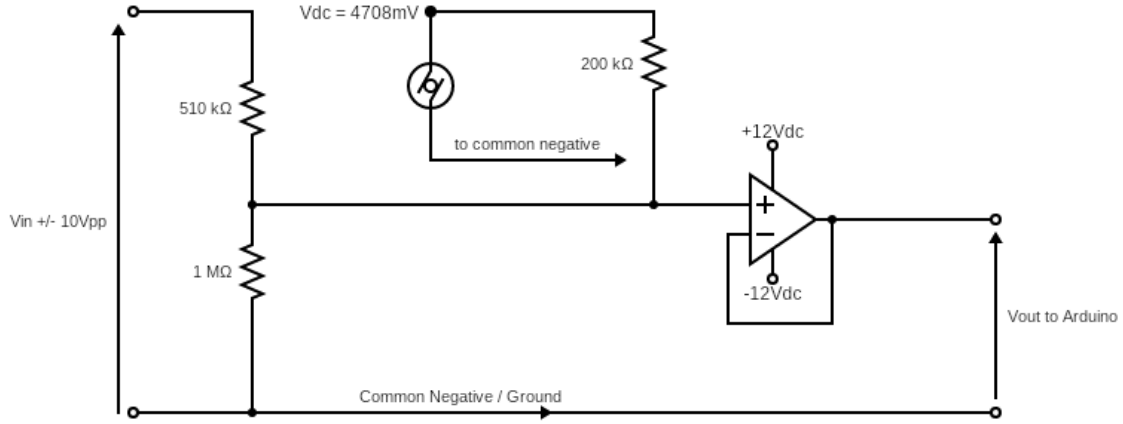

Figure 11 - Acquisition board for a +/- 10Vpp input signal with the TL072CP

I used the Texas Instruments TL072CP amplifier (bought on Amazon). The specification sheet for the TL072CP is available at this link. The amplifier is configured with no gain because I don't need to amplify the signal but need to isolate the output from the input.

I made the voltage divider board and the board with the amplifier using an online calculator for the voltage divider (similar to the previous circuit) and followed the instructions found on the pdf file "Designing Gain and Offset in Thirty Seconds" by Texas Instruments. However, after following the method shown in this paper, I had to tweak the resistances, bias voltage, and amplifier values to ensure I'm within the required range. To do this I used a couple of decade resistors, a scope, and a signal generator.

I found most of the electronic components (such as amplifier socket, resistors, pins, circuit board, solder wire, etc.) on AliExpress and Amazon. However, these can also be found on other outlets.

I also came up with a board to read smaller signals from probes (for example less than 1V) in two configurations shown in Figure 12. In a future project, I will develop these circuits, build the hardware and share the test data.

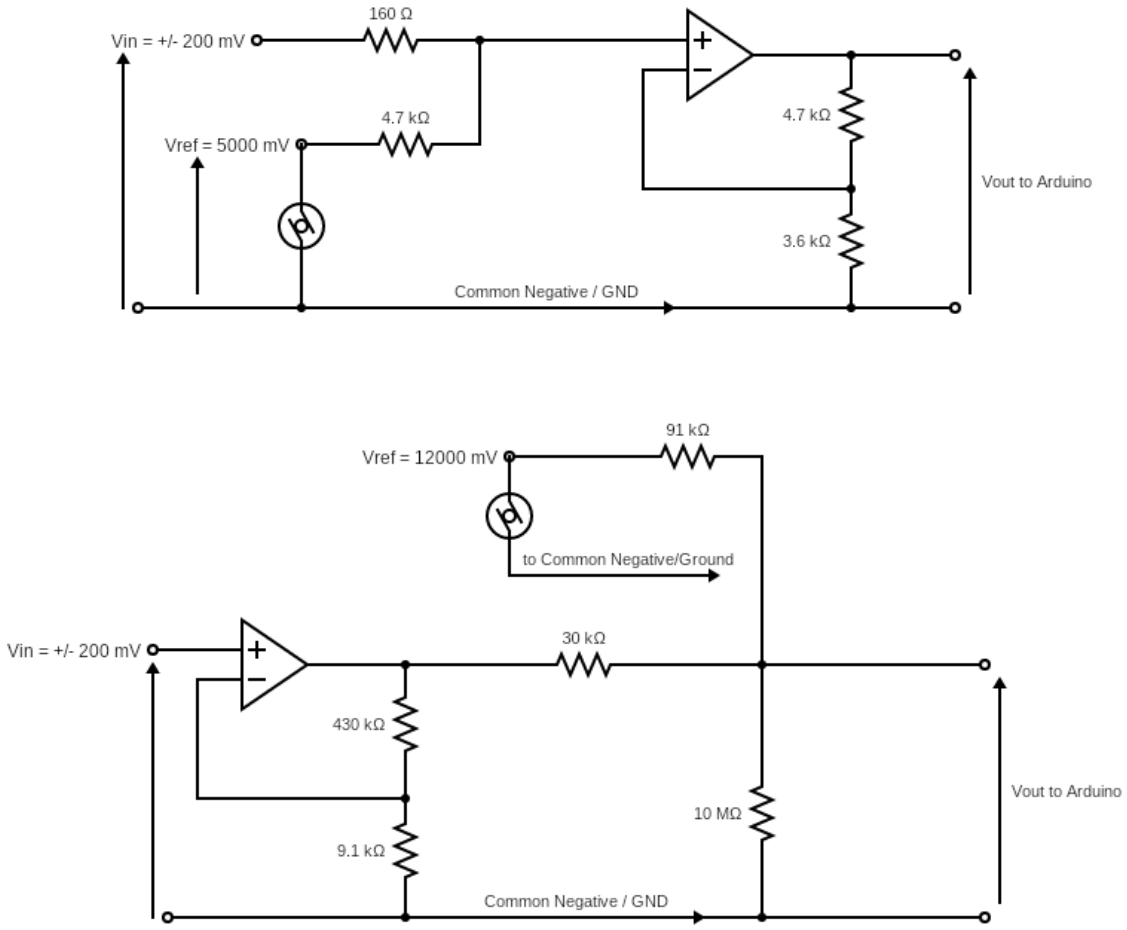

Figure 12 - Step up acquisition boards for a +/- 200mVpp input signal

I attempted to make a filter but didn't proceed because the amplifier configuration shown above filtered all the noise and I didn't find any other sources of interreference (more information on filters in the References section).

The Power Supply Circuits

To power the Arduino, the amplifier board, and other shields, I used the power supply circuit shown in Figure 2.

This circuit is divided in 3 main parts:

1. The power input board - This receives power from the USB source and has multiple output pins that power the Vref power supply and the power distribution board.
2. The power distribution board - This board receives power from the input board and connects to the other power supplies. On this board there is also the dual output (positive and negative) boost power supply for the amplifier.
3. The power supply for the DC bias - This is required to offset the input signal before going through the amplifier.

Why use all these power supplies and why segregate each circuit? Because there are advantages to having multiple power sources:

1. You can use the available power source (power supply or battery) even if it may be at a different voltage of the circuit used. In fact, often the power source available is the most convenient or available. For example, you can use a 6VDC power supply scavenged from an old device and connect it to each data logger circuit's power converter. The latter would boost the power value to the voltage requirements of their respective circuit (amplifier circuit, the Vref circuit, the voltage bias circuit and the Arduino UNO).
2. Each circuit needs a different voltage value. For example, the power for the amplifier will depend on the type of amplifier used: this is usually at a different value than the one required by the Vref and the Arduino UNO. In fact, the amplifier circuit needs positive and negative 12VDC while the Arduino UNO can function at 6 VDC.
3. Separate power supplies allow you to isolate noise: when you loop different circuits that are connected to a common return wire (ground or negative wire), noise can distribute which may alter the circuit behavior.
4. Each circuit needs a different regulation – For example, having different power supplies, allows to change the Vref without affecting the value of the amplifier or the power supply of the Arduino UNO. This is also useful when using additional voltage (or current) acquisition boards such as the AD1015.

For the power supplies, the main parts list is as follows (I bought everything on AliExpress):

- USB input power supply: MT3608 DC-DC Adjustable Boost Module 2A Boost Plate Step Up Module With Micro USB LM2577 Replace XL6009
- Amplifier dual boost power supply: DD1718PA Positive & Negative Dual Output power supply DC DC Step-up Boost Converter module
- Power for the DC Bias: LM2596 DC to DC Buck Converter Voltage Regulator 3.2V-46V to 1.25V-35V Buck Converter Power Supply Step Down Module
- Circuit boards, pins, jumpers, and other accessories are generic.

Power Supply for the Vref

As discussed in the paragraph “Voltage Reference”, the reference voltage must be very stable and ideally with no fluctuations. To achieve this, I used an independent power supply with a TL431, and an output filter as shown on the schematic of Figure 8. The TL431 is the chip that will provide the steady voltage (for details see in the references).

I used the following main parts:

- Power supply: XL6019 (XL6009 upgrade) Automatic step-up step-down Dc-Dc Adjustable Converter Power Supply Module 20W 5-32V to 1.3-35V
- TL431 431 SOT-89 Precision Supply Voltage Reference IC Chip New Original
- Capacitors and resistors: I bought a resistor kit and an electrolytic capacitor kit. These are available on AliExpress, Amazon and similar outlets.

Conclusions and What’s Next?

Overall, I am happy with the sampling rate and precision of the Arduino UNO, but in the future, I plan to investigate other methods that may offer a better ADC resolution.

In presenting the overall Arduino UNO data logger project, I did not go over the details on how to connect the SD card shield, the RTC shield and the other components. However, I’m sharing all my sources of information, pictures, schematics, and diagrams.

I know that there are other topics to cover, but I just couldn’t do it all in one paper.
In fact, I need to investigate the overall precision of the measurements that include the extra electronic boards that I’ve shown in Figures 10, 11 and 12. I plan to discuss this in a future paper.

Additionally, I was presented with a new challenge of how to visualize the data upon completion of the data logger. The sources I found online didn’t provide the code that would fit my needs, so I created my own scripts on Python (I found all the information on StackOverflow). I ran the Python on Jupiter Notebook (running on Anaconda). I plan to discuss this work and results in a separate paper.

One last thing: don’t forget to look at my Reference section to find more interesting videos, projects, and information.

References

Below I share various sources that helped me in understanding Arduino and creating this paper. I started with my go-to sources I depend on to find information and the rest are other useful sources.

DroneBot – Website and YouTube channel with various Arduino projects. This is my number one go to source that I consult before starting a new project.
Andreas Spiess – YouTube channel with projects on Arduino and other microcontrollers
MJLorton – YouTube channel with excellent tutorial and instructional videos explaining how things work and going through a variety of projects with Arduino and electronics
EEVBLOG – Website and YouTube channel: Just about everything on electronics, instruments, various devices and has many tutorials on a variety of projects.
GreatScott! – YouTube channel with a variety of projects with Arduino and electronics
W2aew – Excellent teacher of Electronics, how to use a scope and sampling rate
Learn Electronics – Several electronic projects with and without Arduino.
Kiss Analog – Good source to check for instruments and simulators.
Ralph S Bacon – YouTube channel with an infinite number of videos on electronics and microcontrollers. I find these extremely useful to watch when I need to master a new topic. He really goes in details and explains everything in detail.
How To Mechatronics - YouTube channel with a variety of projects with Arduino and electronics
Julian Ilet – Excellent source of inspiration for a variety of electronics projects.
Kris Kasprzak - YouTube channel with a variety of Arduino projects (and not just Arduino).

Also, a great book that teaches Arduino through fun data logging experiments is "Exploring your Environment with Arduino Microcontrollers – by David R. Brooks PHD, Institute for Earth Science Research and Education, 2020. This is the only book that I found that focuses on data logging.

In the following I provide references based on each topic discussed.

Arduino General Info

Programming Arduino Nano Using UNO
How to access multiple SPI interfaces on Arduino

YouTube Videos

#20 Tutorial: Multiple Devices on One Arduino I2C Bus
Arduino Fix Error Low memory available, stability problems may occur

Timing the Arduino Code and Timing Events (with Internal Timers):

Using Millis for Timing
Clovis Fritzen Github Codes

YouTube Videos

Arduino Programming Tutorials || How to Construct Time-Sensitive While Loops
TUTORIAL: How to Manage Arduino Timing: Delay, Millis & Micros!
Electronic Basics #30: Microcontroller (Arduino) Timers
#209 Arduino Timer Interrupts - Overflow & Comparator. Easy Peasy.
Arduino Power Saving Tutorial: Clock Pre-Scale and Sleep

Timing Events with Date and Time

DateTime, values, and multiple comparison operator (Not sure I verified this code)
How do I create an if statement based on RTC time?
Arduino Turn on / off anything at a specific time (Trigger a Relay with DS3231 RTC)

Arduino Shields

https://learn.adafruit.com/adafruit-data-logger-shield
https://www.play-zone.ch/en/dk-data-logging-modul.html

SD Card

Data Logger RTC by Deek Robot - Got to verify this source, I don't remember if I used it.
https://github.com/PaulStoffregen/SD/blob/master/examples/Datalogger/Datalogger.ino

For longer file names, I used the SDFat.h library from Adafruit. Read more about this in the next three blogs:

- Do you want Long File Names in SdFat?
- How to use long file names on Teensy/Arduino SD Card Reader
- SdFat with Long File Names - smaller than SD.h with 8.3 names

YouTube Tutorial Videos

Using SD Cards with Arduino - Record Servo Motor Movements
Arduino Tutorial: SD card module Micro SD tutorial DIY.

Real Time Clock (RTC)

Arduino and DS3231 Real Time Clock Tutorial
RinkyDinkElectronics DS3231 Library
https://adafruit.github.io/RTClib/html/class_r_t_c___d_s3231.html
Real Time Clock: Setting the Date/Time with Arduino

Arduino Displays

YouTube Videos

- Displays and Arduino.
- LCD TFT

Arduino Analog Pins and ADC Resolution

Dividing by 1023 or 1024? The final verdict on analogRead
Is there a limit on how much current a pin can sink?
Maximum Current/Voltage into an analog pin on an Arduino UNO
How to change the ADC resolution
Read Positive and Negative Voltage in Arduino [closed]

YouTube Videos

Maximizing Arduino's ADC Resolution and Accuracy Part 1
Electronic Basics #27: ADC (Analog to Digital Converter)
Arduino Tutorial #5 - Digital Voltmeter, Arduino Analog to Digital Converter
ADC blocks and REGISTERS | Internal Reference | Internal Temperature Sensor Arduino101
Measure up to 500A DC Current with Shunt Resistor using Arduino

Sampling Rate

How to change the ADC resolution
#96: Tutorial on Digital Oscilloscope sample rate, record length and data processing

Online Calculators

Voltage Divider Calculator

Measuring Voltages and Currents with Arduino

Read Positive and Negative Voltage in Arduino [closed]
How to log voltage and current?
John Errington's Experiments with an Arduino:

- Voltage measurement with the Arduino board: Logging data
- Voltage measurement with the Arduino board (cont).

YouTube Videos

#347 Measuring Mains Voltage, Current, and Power for Home Automation
Arduino Tutorial #5 - Digital Voltmeter, Arduino Analog to Digital Converter
#189 Zero Crossing Detection using an Arduino (TRIAC Dimmer Control)
MPPT Solar Charge Controller #3 - New Display, Watts Peak and Bargraph
Arduino based current meter with i2c LCD screen
TUTORIAL: How to Measure Current - Arduino - Current Shunt & Amplifier (Part 2 - Wireup & Code)
Measure up to 500A DC Current with Shunt Resistor using Arduino
Make Digital Voltmeter Ammeter Voltage Current Meter With JLCPCB
Easy measure of any AC voltage with Arduino and ZMPT101B (up to 250V)
Homemade DC/AC Oscilloscope Current Clamp | Hall Sensor + Ferrite Core
Arduino based current meter with i2c LCD screen

Vref - Arduino Voltage Reference

Making Accurate ADC Readings on the Arduino

YouTube Videos

#9 Arduino Data Logger with Direct Input to Excel

#10 Tutorial: Make the Arduino Analog Readings more precise
#340 How good are the ADCs inside Arduinos, ESP8266, and ESP32? And external ADCs (ADS1115)
Utilizing the Arduino ADC Internal Reference
Arduino Basics: analogReference()
Arduino Tutorial #5 - Digital Voltmeter, Arduino Analog to Digital Converter
ADC blocks and REGISTERS | Internal Reference | Internal Temperature Sensor Arduino101
T4D #2 - Voltage References, calibration, accuracy, resolution..

YouTube Videos and Papers about the TL431

TL431, TL432 Precision Programmable Reference (PDF specification Sheet)
TL431 Based Current Limiter Constant Current Source Circuits
TL431 Secrets and Pitfalls
Part1:tl431 precisely,very accurate linear voltages regulator
Make an adjustable precision Zener diode
How Does TL431 Work in an Isolated Flyback Supply
How to TEST TL431 Voltage Reference / TL431A TL432 KIA431 Shunt Regulator circuit

Power Supplies (to Power Arduino and for the Vref)

Power For Your Electronics Projects - Voltage Regulators and Converters
#334 How to find the right Power Supply for your Project
Dual Voltage Supply build with the ICL7660
How Does a Computer Power Supply Work (ATX)
Arduino Tutorial: Using external voltage reference
Maxim MAX6350 Voltage Reference

Arduino Data Logger Projects that Helped me

Arduino Datalogger for Vernier Conductivity Meter and Peristaltic Pump Controller
Arduino Alog Data Logger Toolkit
Voltage and Current Data Logger

YouTube Videos

Arduino SD Card and Data Logging to Excel Tutorial
Arduino Project: Advanced Datalogger with ATMEGA328, BMP180, DHT22, BH1750 and sd card
Arduino BMS #7: Adding A Voltage Reference and Input Shutoff (This is one video of a series made by Eman2000)
Solar & Wind Data Logger
#53 Protect your Arduino - use an Opto Isolator! And RGB sound-to-light demo

Other Arduino Data Loggers

AC Current Monitoring Datalogger
AC Voltmeter Using Arduino
Datalogging 6 analog inputs with Uno… (Not sure I verified this source, not sure the code works)
Logging Shield - Data Logging for Arduino

About the TEENSY and Programming the Registers

Check out Matt Bilsky's YouTube channel with very interesting projects.

https://github.com/LAtimes2/DataLogger
https://forum.pjrc.com/threads/43708-Teensy-3-6-Datalogging-at-10kHz

Clovis Fritzen
https://hackaday.io/project/5958-teensy-data-logger
Clovis Fritzen Github Codes

YouTube Videos

Easy & Powerful Arduino Alternative? #3 Teensy Beginner's Guide
Build a Teensy-based thermal imaging camera

About Operational Amplifiers

Measure both positive and negative voltages using ADC
Scale 30-50 mV signal to 0-5 V range
GCE Electronics – Chapter 4: Operational Amplifiers - Link to PDF to Download
"Designing Gan and Offset in Thirty Seconds" by Texas Instruments

YouTube Videos

TUTORIAL - Build TL072CP Circuit On Breadboard (Easy-Medium Difficulty)

Reducing Noise During Measurements

Reduce Noise in Your Sensor Measurements with an Active Low Pass Filter Part 1
How To Reduce ADC Noise Through Filtering Analog Inputs

Projects by upgrdman to visualize data (YouTube Videos):

- GPU-Accelerated Arduino Data Logging and Telemetry
- Easy Arduino Data Logging and Telemetry (Project to visualize data with arduino)
- Easy WiFi Telemetry with an Arduino and ESP8266
- Advanced Arduino Telemetry and Data Visualization