MTR-Duino Engineer’s Guide (Rev 5.2)

Table of Contents

[1. Design Decisions 2](#_Toc501450085)

[1.1. Arduino Micro 2](#_Toc501450086)

[1.2. RTC 2](#_Toc501450087)

[1.3. Batteries and Power Boost Converter 2](#_Toc501450088)

[1.4. Temperature Measurement Circuit 3](#_Toc501450089)

[1.4.1. Preventing Self Heating 4](#_Toc501450090)

[2. Circuit Description 5](#_Toc501450091)

[2.1. Arduino Micro 5](#_Toc501450092)

[2.2. Power and Boost Converter 5](#_Toc501450093)

[2.3. microSD Card and RTC 5](#_Toc501450094)

[2.4. Temperature Measurement 6](#_Toc501450095)

[3. Programming 7](#_Toc501450096)

[3.1. Program Flow 7](#_Toc501450097)

[3.2. Uploading Firmware (Reprogramming) 8](#_Toc501450098)

[4. Building New Instruments 15](#_Toc501450099)

[4.1. Printed Circuit Boards, Components, and Assembly 15](#_Toc501450100)

[4.2. Thermistor Mounting 15](#_Toc501450101)

# Design Decisions

## Arduino Micro

The Arduino Micro was used in the design of this instrument primarily to simplify the design process (that and the principal engineer was not formally trained in electronics, thus had to make some accessions). The Arduino platform is an open-source development platform designed, which combines a micro-processor, some fundamental software libraries, and a programming environment to make it very easy for newcomers to develop electronics equipment. Since the Arduino is open-source, there are volumes of information online about the specifics which will not be discussed here. The Arduino Micro is one of many Arduino boards and was chosen because a) it fits in the MTR cases and b) it uses a simple USB connection.

One main drawback of using the Arduino platform is power consumption. When the MTR-Duino is sampling it draws ~70mA. Other simple oceanographic instruments often draw less than 10 mA. This is still tolerable, because the instrument only needs to be on for a short time (2 seconds) to take a single discreet sample before going back to sleep.

One other major hurdle was that the Arduino Micro does not have a ‘low-power’ mode. In order to overcome this problem, the Arduino is completely shut down when not sampling. As the Arduino cannot ‘wake itself up’, a RTC with a built in alarm is used to send a signal to a power converter and wake the unit. This is not a traditional or even optimal design, but it does work for the application, keeps cost down, and has proven a reliable method in several other projects (TAPS controller and Pop-up Buoys).

In the future, it may be desirable to switch the design to a standard processor used for oceanographic instruments. This would have a few main advantages – a) lower power consumption when sampling, b) smoother communication with terminal programs (Arduinos can be a bit finicky), and c) a more standardized instrument which doesn’t require unique software or equipment. PMEL’s EDD should be consulted or included if choosing to this approach in order to optimize and standardize the equipment as best as possible.

## RTC

The MTR-Duino uses a DS3234 RTC with an internal alarm to ‘wake up’ the unit from sleep at prescribed sample intervals. This is done because the Arduino does not have an on-board clock to do so and must be shut down to conserve power. The alarm is set within the program using the appropriate registers according to the DS3234 datasheet.

## Batteries and Power Boost Converter

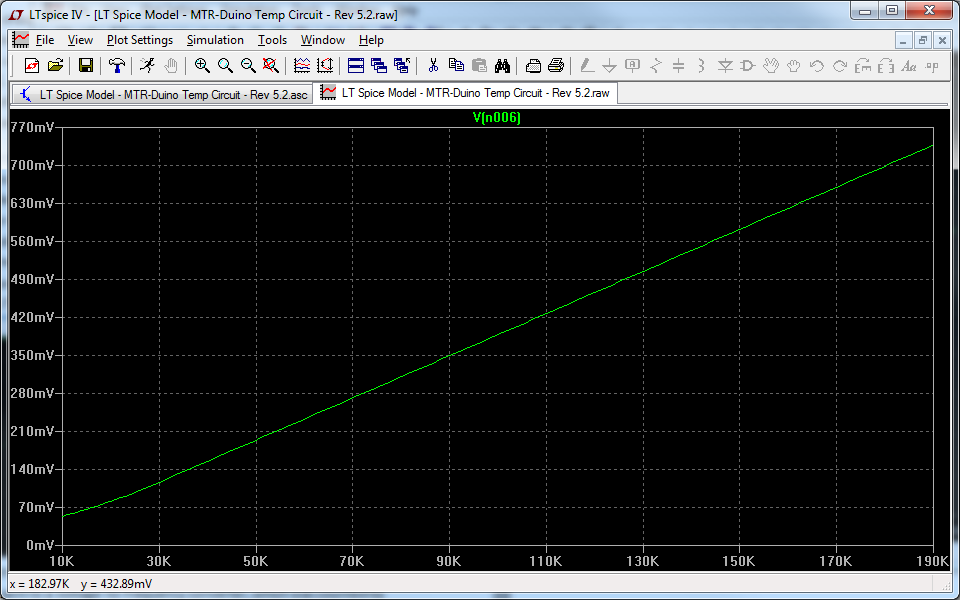
One of the unique features of the MTR-Duino circuit is a power boost converter that regulates voltage to the board and allows it to wake up and go to sleep. It is based off Adafuit’s PowerBoost 500 Basic module. The TPS61090 at the heart of boost converter is capable of producing the 5V voltage needed to run the Arduino Micro from a primary battery voltage as low as 1.8V.

It may seem odd to boost the voltage rather than use a higher voltage battery, especially when the voltage is again stepped down for the other ICs on board, but in this scenario it makes quite a bit of sense. The TPS61090 operates at 90%+ efficiency and provides the ability to fully drain the batteries. This allows more power to actually be drawn from two AA batteries as would be drawn from a single 9V battery. (Due to space constraints, those are the two best options).

The enable pin on the TPS61090 is also what allows the Arduino Micro to be completely shut off and then woken with a signal from the RTC. The dual advantage of this circuit makes it a sensible design. If this instrument were to be re-designed using a standard oceanographic micro-processor one day, this portion of the circuit would not be necessary.

## Temperature Measurement Circuit

The temperature measurement circuit used in this design was adapted from the original MTR. It was altered slightly to produce a smaller amount of current through the thermistor (to reduce self-heating effects) and produce a broader voltage output range (to use the full measurement range of the ADC more effectively.) The circuit produces a voltage output which is perfectly linear for temperatures in typical oceanographic applications, and only slightly non-linear for high temperature applications (above 35°C). The graph below shows the voltage output of the analog portion of the instrument as the resistor varies from 10k to 190k (approximately 66°C to -3°C).See the LT Spice Model for exact voltages and output linearity. The thermistor is 50kat 25°C and increases in resistance at lower temperatures. This means that the instrument is perfectly linear and has higher resolution at lower temperatures where it is used most frequently.



The original MTR used a circuit with a shunt regulator and operational amplifier connected to the thermistor, which produced a voltage sent to a voltage-to-frequency converter, which was counted by the micro-processor. In the years since the original design, electronics have seen some marked advances in many areas – one of them being with Analog to Digital Converters. The ADS1100 ADC used in the MTR-Duino is a self-calibrating ADC capable with 0.02% accuracy. This is just *slightly* less than the 0.01% accuracy needed to achieve the 0.01°C design requirement, so a 0.01%, 0.2ppm reference resistor is used not only to track any drift in the sensor, but also to correct for any inaccuracy in the ADC.

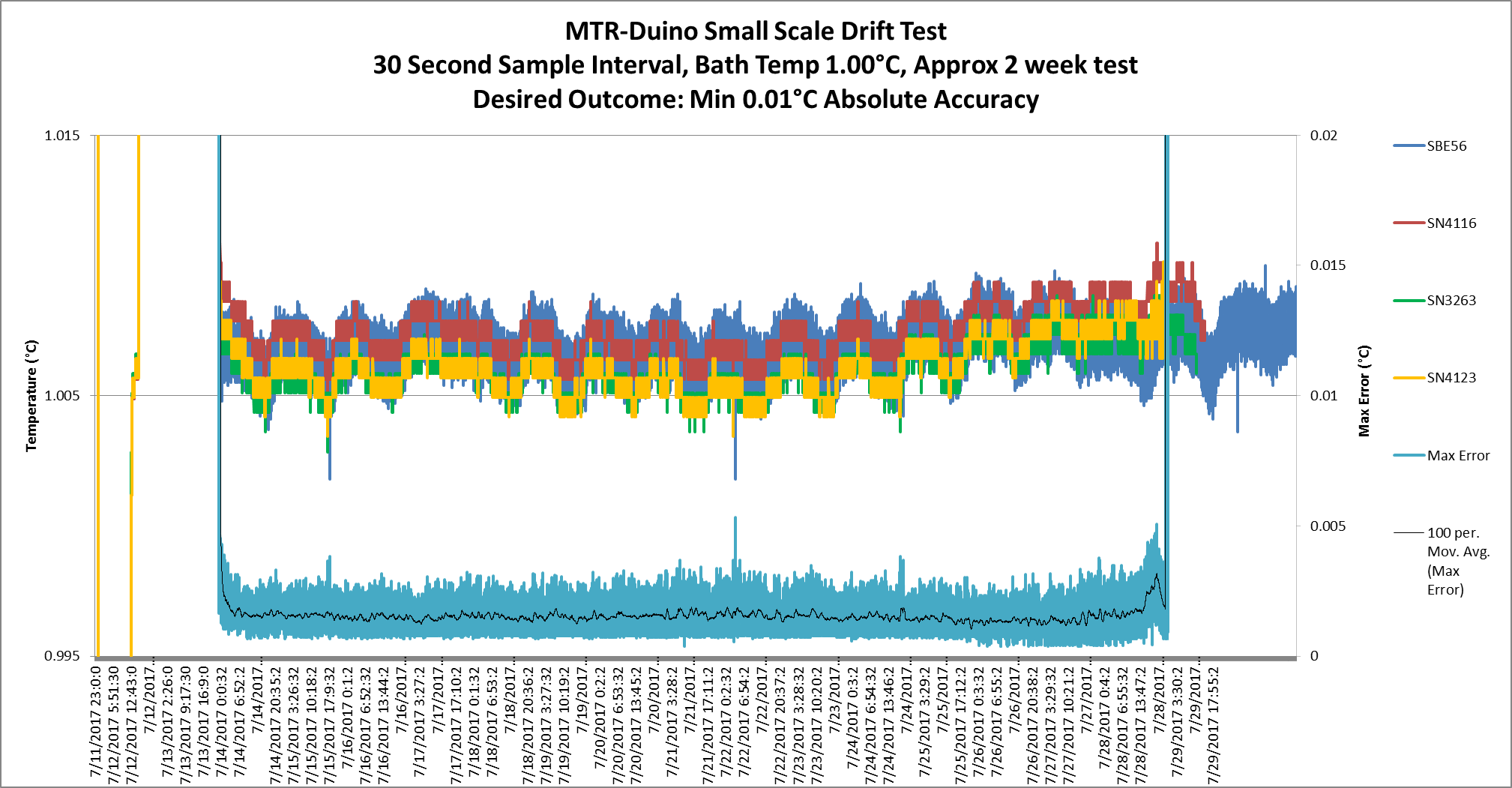
### Preventing Self Heating

One of the most difficult problems to troubleshoot and correct in this design was an issue caused by thermistor self-heating. When a current flows through a thermistor, it will generate heat which will raise the temperature of the thermistor above that of its environment. For a typical thermistor, only ~10A of current is enough to raise the temperature of the thermistor by 0.01°C in still air.

To reduce any self-heating effects to less than 0.01°C during calibrations of this instrument, it was necessary to increase the resistance of the thermistor to 50k and sample at a maximum interval of 30 seconds. (Changing the thermistor resistance from the original circuit also required changing 3 other resistors in the circuit – see the LT Spice model for full details).

Below is a graph showing 3 calibrated MTRDuinos along with an SBE56 for comparison, resting in a 1.00°C bath for 2 weeks. The error is well below the 0.01°C required for this device for the entire duration.

**In order to ensure self-heating effects do not affect the MTRDuino’s calibration or data, the sample interval should never be set to less than 30 seconds, during calibration or deployment**.



# Circuit Description

## Arduino Micro

The heart of the MTR-Duino is an [Arduino Micro](https://www.arduino.cc/en/Main/ArduinoBoardMicro) which controls read and write functions to the Real Time Clock, microSD Card, Analog to Digital Converter, and temperature reference switch. It uses both I2C and SPI communications as well as digital IO signals (HIGH/LOW).

## Power and Boost Converter

Power is regulated and switched on and off with a [TPS61090 Synchronous Boost Converter](http://www.ti.com/lit/ds/symlink/tps61090.pdf) (U1). The design here was taken exactly from [Adafruit’s PowerBoost500 module](https://www.adafruit.com/products/1903?gclid=CMS71dH52b8CFc5afgodggIAIg) (schematic also contained in documentation). The circuit provides a steady +5.2V output from a battery as low as 1.8V with ~90% efficiency. In this case, the primary battery (B1) provides ~3V to the boost converter under normal conditions. Any disadvantage from efficiency lost in the boost converter circuit ramping up the voltage is overcome by the fact that more energy can be drawn from the batteries until they are fully drained.

R3 through R6 set the voltage output level at 5.2V. Both the RTC and Arduino Micro are connected to the Enable pin on the Boost Converter. When waking up, the RTC alarm pulls the Enable signal high, turning on power to the unit. The Arduino Micro then sends a high signal via the ‘Shutdown’ Pin which keeps the unit running while the alarm is reset and sampling is completed. Eventually, the Arduino Micro sends a low voltage signal via the ‘Shutdown’ pin, forcing the Enable signal low and shutting off power to the unit. R2 is a pull down resistor to ensure the Enable signal is normally low.

3.3V power is needed for the RTC, microSD Card, and I2C communication with the ADC and is provided by U12, an [LP5907 Ultra-Low Quiescent Current LDO](http://www.ti.com/lit/ds/symlink/lp5907.pdf).

## microSD Card and RTC

The RTC and microSD Card require SPI communications at 3.3V and the Arduino Micro operates at 5V. U10, a [MC74VHC1GT125 Non-Inverting Hex Buffer](http://www.ti.com/lit/ds/symlink/cd74hc4049.pdf) which accepts over voltage inputs, is used to lower voltages coming from the Arduino (SCK, MOSI, CS) and U8, a [NC7S14 Single Non-Inverting CMOS Logic Level Shifter](http://www.onsemi.com/pub_link/Collateral/MC74VHC1GT125-D.PDF), is used to raise the MISO signal going to the Arduino Micro.

U9 is a standard [microSD Card Push-Push Socket](http://www.molex.com/pdm_docs/sd/473340001_sd.pdf) to allow the user to remove the card and download data quickly.

U2 is a [DS3234 Extremely Accurate Temperature Compensated RTC](http://datasheets.maximintegrated.com/en/ds/DS3234.pdf) which provides both timekeeping function and an alarm to wake the unit up and collect samples at the proper time. B2, a 12mm lithium coin cell battery, powers the RTC when the unit’s primary power is off. U3, an [NC7S14 High Speed Inverter with Schmitt Trigger Input](https://www.fairchildsemi.com/datasheets/NC/NC7S14.pdf), is used to invert the signal coming from the RTC alarm to be the proper voltage for the Enable input of the Boost Converter.

When the alarm function on the RTC is not active, the INT/SQW pin is held at high impedance. This keeps the input signal to U3 at high voltage (3V), and the output to the Enable pin low (power off). R8 is a pullup resistor to ensure the input signal to U3 is normally high. When the alarm function is triggered, the INT/SQW pin on the RTC drops to low impedance, allowing current to flow and driving the input of U3 to low voltage. This causes the output of U3 to be high voltage (3V) and the high signal sent to the Enable pin turns power to the unit on. U3 and R8 are tied to the primary battery, rather than to the coin cell, to prevent the coin cell battery from draining if the alarm is triggered, but the primary battery is dead or not connected. After all, the alarm function is only needed if the primary battery is available.

D1 and D2 are Schottky Diodes that ensure the shutdown signal coming from the Arduino and shutdown signal coming from U3 do not cause a short. The low forward voltage of the Schottky (450 mV) keeps the voltage at the Enable pin at the highest possible voltage to ensure the Boost Converter trips on.

## Temperature Measurement

The actual voltage measurement in the unit is done by U4, an [ADS1100 Self-Calibrating 16-Bit ADC](http://www.ti.com/lit/ds/symlink/ads1100.pdf), which achieves approximately 0.02% absolute accuracy using the settings in this instrument. The ADC requires a highly accurate power source, provided by U7, a [MAX6126 Ultra-High-Precision Voltage Reference](http://datasheets.maximintegrated.com/en/ds/MAX6126.pdf).

The ADC requires I2C communications at ~3V and the Arduino Micro operates at 5V. Q2, Q3, and R12 through R15 are used provide bi-directional I2C communications at the proper voltages. The I2C communications are tied to 3.3V rather than 3V to keep the ADC on an isolated circuit and because the ADC/communication protocol can tolerate the small difference in voltage.

The analog portion of the circuit was adapted from the original MTR design. It provides an almost perfectly linear voltage output between 0.05V and .75V when the thermistor is between -5°C (~190k) and +65°C (~10k). The non-linear portion is only at higher temperatures where higher accuracy is not normally needed in oceanographic applications. (See the LTSpice model for more specific info). After the voltage signal is output by an [LT1013 Precision Operational Amplifier](http://www.ti.com/lit/ds/symlink/lt1013.pdf), a gain of 4 is applied internally by the ADC before sending the digital signal to the Arduino Micro. During calibration, the voltage measurement is directly fit to a Steinhart-hart equation, so conversion to resistance or any other values is not needed.

A 0.1°C interchangeable NTC thermistor is used, which can provide 0.1°C accuracy by using calibration coefficients from a different unit. This is advantageous if calibration is not possible for some reason, or calibration info is lost or corrupted. On the Printed Circuit Board, the analog circuit elements are on a separate isolated ground plane, linked to the main ground plane only by one small trace. This is done to reduce any potential ground noise from digital signals in the rest of the circuit.

Since 0.01°C accuracy was needed for this application, but the ADC and voltage reference can only provide ~0.02°C accuracy (~0.02%), R11, an ultra-high precision reference resistor with 0.01% tolerance and ±0.2ppm temperature coefficient, is used to correct for any inaccuracy. U5, an [ADG836 Dual SPDT Switch](http://www.analog.com/media/en/technical-documentation/data-sheets/ADG836L.pdf), is used to swap out the thermistor and reference resistor in order to take the alternate measurement. Low on-state resistance and tight channel-to-channel matching make this switch optimal for providing consistent measurements. The resistors are swapped out by a single digital signal sent from the Arduino Micro which is translated to 3.3V by U11, a [NL17SZ07 Single Non-Inverting Buffer with Open Drain Output](http://www.onsemi.com/pub_link/Collateral/NL17SZ07-D.PDF). R16 is a pullup resistor to make sure the output voltage is normally high and does not float.

# Programming

## Program Flow

The general program flow when taking a sample is to do the following:

1. Wake Up

2. Initialize Pins and Libraries

3. Initialize ADC

4. Get Time from RTC, Reset RTC Flags, Set Next Alarm to wake up unit

5. Get voltage reading for thermistor from ADC, Store to microSD Card

6. Send digital signal to switch from thermistor to reference resistor

7. Wait 500 milliseconds to let voltage levels stabilize

8. Get voltage reading for reference resistor from ADC, Store to microSD Card

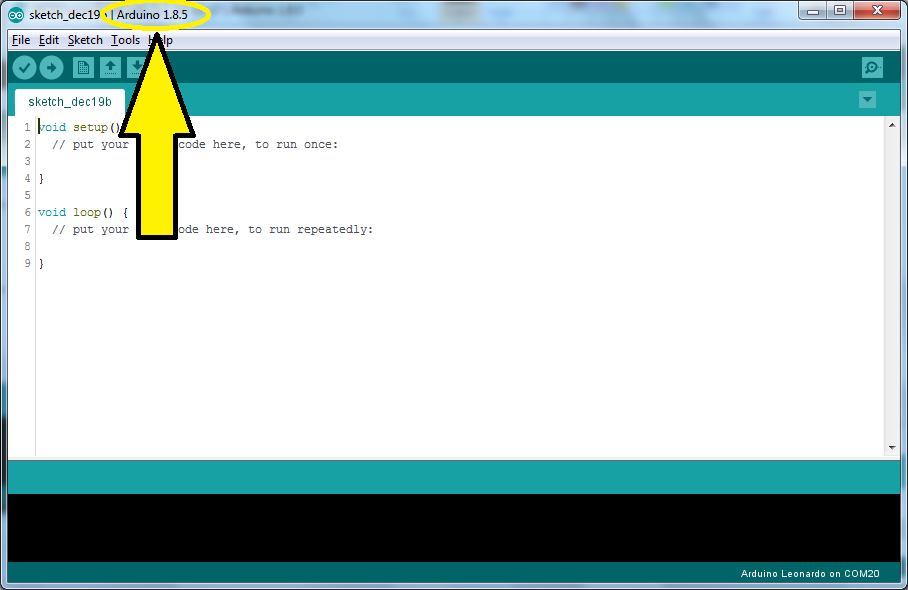
9. Send digital signal to boost converter to shutdown unit.

## Uploading Firmware (Reprogramming)

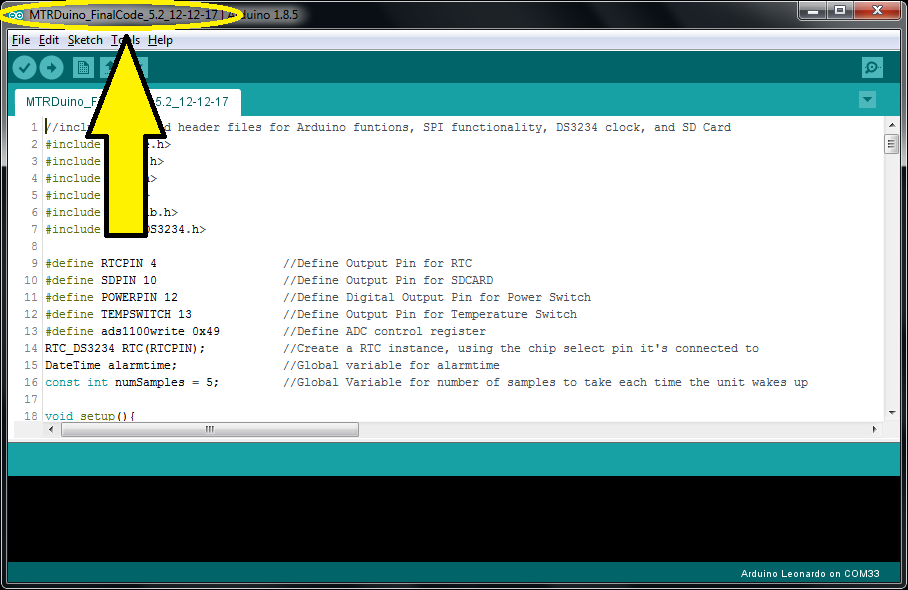
The Arduino-Micro (ATmega32u4) is programmed and firmware is installed with its own program, the Arduino IDE (Version 1.8.5). This should not need to be done under normal circumstances. However, if the firmware is updated, new units are built, or the firmware is corrupted for any reason, it may need to be re-installed.

To install the firmware:

1. **Open the Arduino IDE (Integrated Development Environment). Be sure to use version 1.8.5.**

**

1. **Open the sketch containing the correct version of the MTR-Duino code**

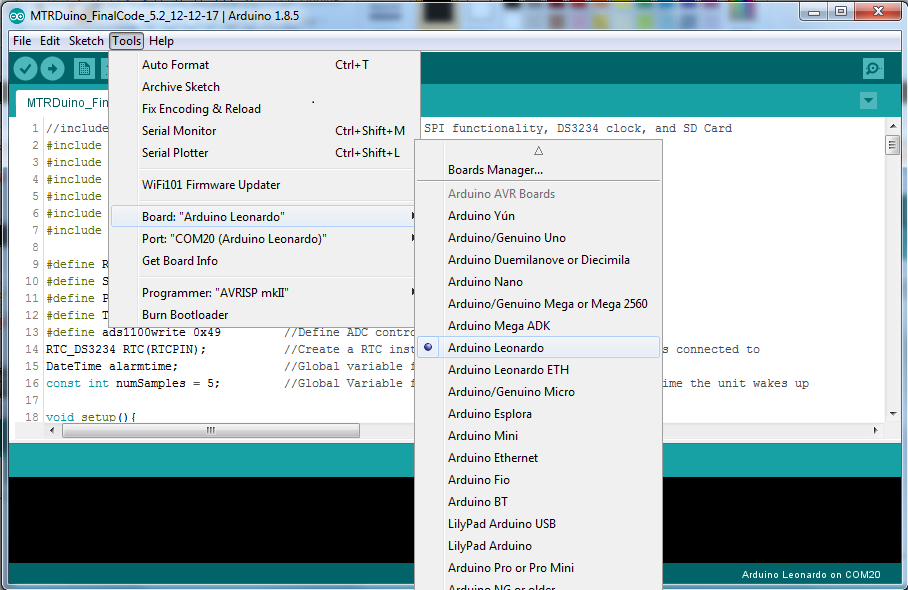


1. **Connect the Arduino Micro**

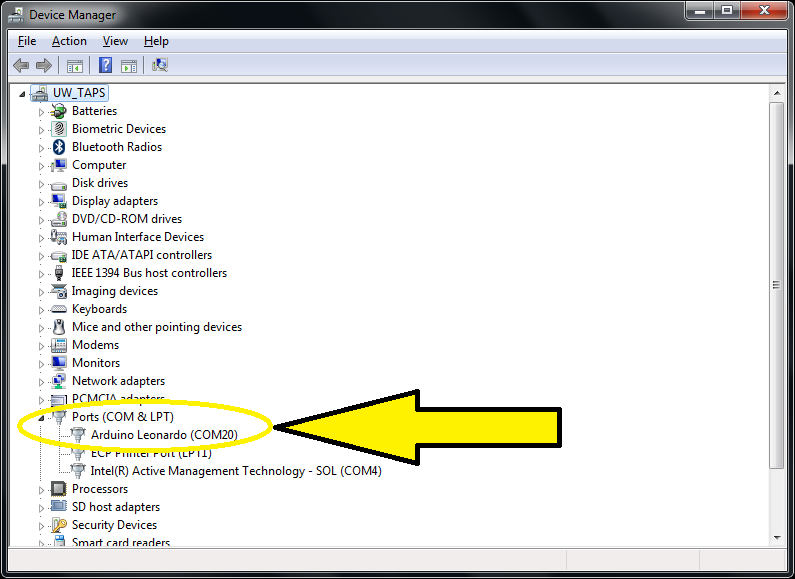
* Remove the Arduino Micro from the Instrument
* Connect the Arduino Micro to the computer with a Micro-USB Cable. You should see a steady yellow LED lit on the top of the Arduino Micro and a steady Blue LED lit on the underside of the Arduino Micro.
* Use only Anker brand or other high quality USB Cables.

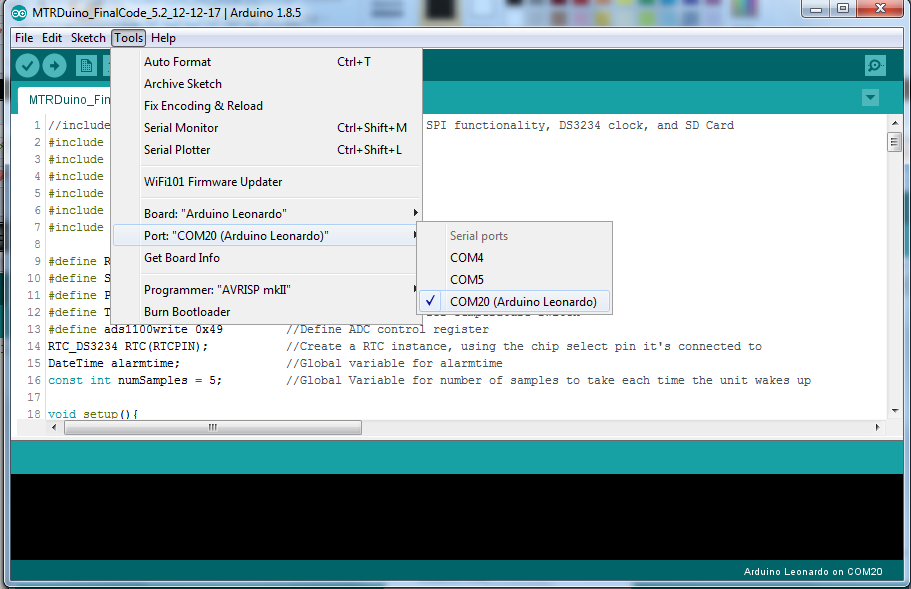
1. **Select “Arduino Leonardo” as the Board. (*Tools 🡪 Board 🡪 Arduino Leonardo)***



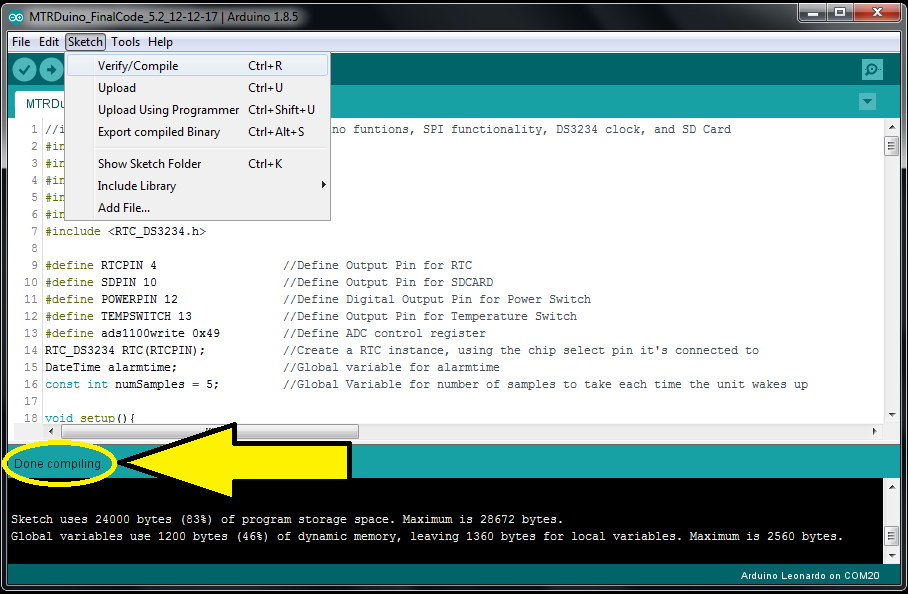
1. **Check which COM port the Arduino board is mounted as. (*Device Manager 🡪 Ports*)**



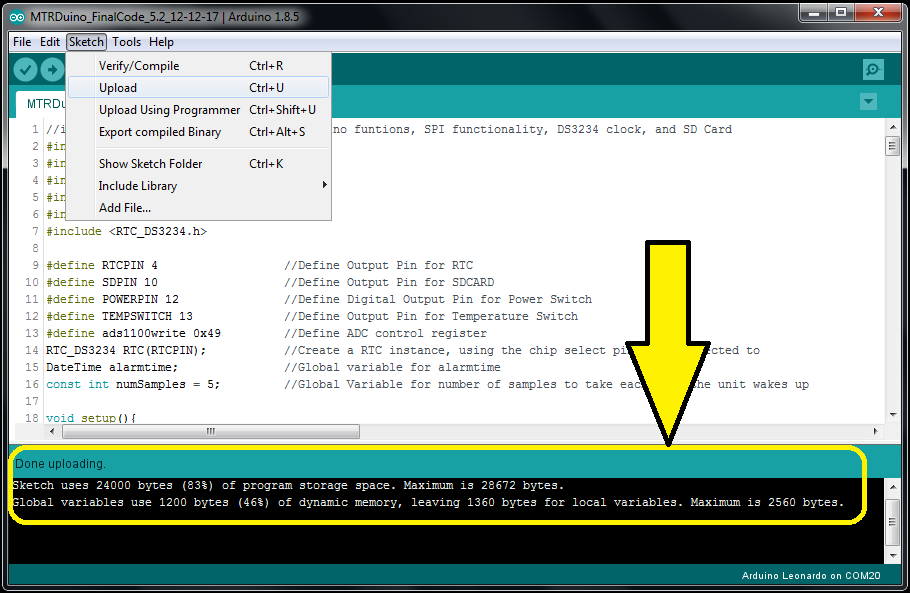
1. **Select the proper COM Port. (*Tools 🡪 Serial Port 🡪 COM \_\_\_*)**



1. **Compile the sketch to make sure there are no errors (*Ctrl +’R’* or *Sketch*** *🡪* ***Verify/Compile*)**



1. **Upload the sketch to the board (*Ctrl +’U’* or *File 🡪 Upload*)**



*\*\*\* If there are any issues uploading the code, but it compiled properly:*

* *Close IDE and Terminal Windows*
* *Unplug the USB cable*
* *Restart Computer*
* *Start at Step 1*

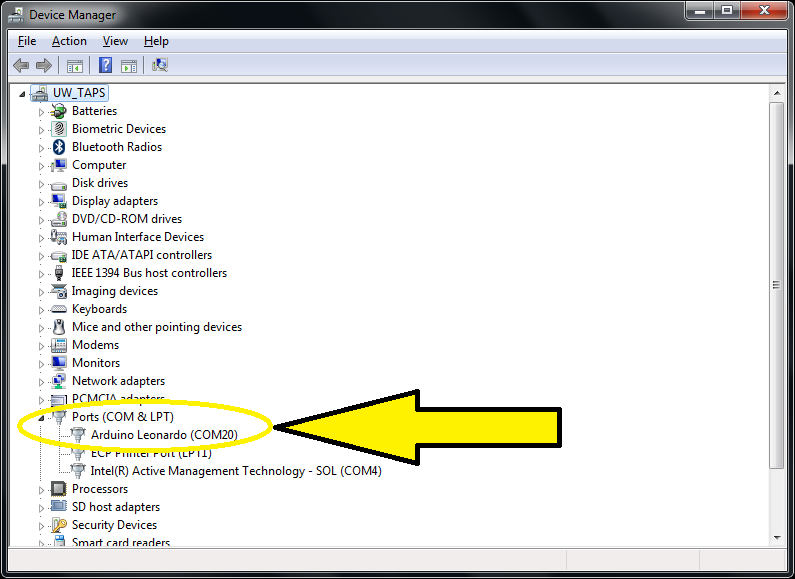
*There are occasionally issues with the Virtual COM Port when there are serial communication errors causing the Arduino to mount as a different COM Port and install new drivers. In this case the code will not upload properly. The easiest solution is to close everything and restart the computer.*

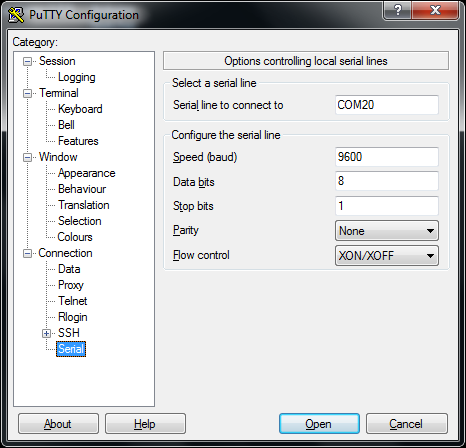
1. **Re-connect the Arduino Micro to the Instrument**

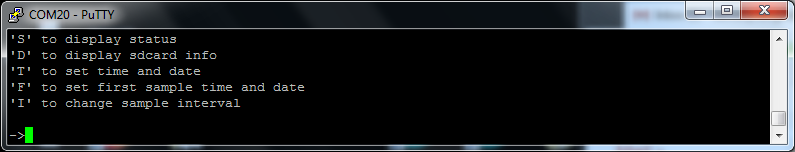
* Unplug the USB Cable
* Insert the Arduino Micro into the Instrument
* Connect the Arduino Micro to the computer with a Micro-USB Cable. You should see a steady yellow LED lit on the top of the Arduino Micro and a steady Blue LED lit on the underside of the Arduino Micro.
* Use only Anker brand or other high quality USB Cables.

1. **Open a terminal program (Putty) using the settings below to ensure code has uploaded properly.**
   * COM Port: Check “*Device Manager🡪Ports*” to find correct COM Port Number
   * Baud Rate: 9600
   * Data Bits: 8
   * Stop Bits: 1
   * Parity: None
   * Flow Control: None







1. **The instrument is now ready to use. Refer to the MTRDuino User’s Guide for instructions.**

# Building New Instruments

## Printed Circuit Boards, Components, and Assembly

The main PCBs for the MTRDuino have been designed using EAGLE. A full list of components is provided in a BOM for the PCB. Assembly of the PCBs has been done reliably by [Schippers and Crew](http://schippersandcrew.com/) for all past revisions

The Arduino Micro boards can be ordered from a variety of suppliers, however; many counterfeit Arduino devices exist on the market – [the purchaser should ensure the boards are genuine](https://www.arduino.cc/en/Products/Counterfeit) and made by Arduino in Italy.

The majority of other components can be easily acquired through [Digikey](https://www.digikey.com/en/resources/homepage) or a similar electronics distributor. The Bill of Materials is in a format that can be quickly edited and uploaded to Digikey to build a cart with the appropriate number of components.

Component L1 (a 6.8 H inductor) is used in the power boost portion of the circuit, which was modeled after [Adafruit’s PowerBoost500 module](https://www.adafruit.com/products/1903?gclid=CMS71dH52b8CFc5afgodggIAIg). L1 is now an obsolete component an exact replacement could not be found when it became obsolete. 500 of these were purchased to be kept for future devices and should be kept as long as possible.

## Thermistor Mounting

Before the MTR-Duino is installed into the pressure case, the thermistor should be mounted using a thermal epoxy. This will help transfer heat from the pressure housing directly to the thermistor, reducing the thermal time constant of the instrument. Thermal epoxy has all the advantages of thermal paste (better thermal conduction, no electrical conduction) without the mess, since it hardens completely. The only disadvantage is that it makes installation permanent, so components should be installed carefully.