# **Electronics Guide**

**What Is This?**

A **resource for anyone interested in working with electronics**. This guide will help you navigate the world of electronics.

* Topics start with [shopping, learning, and practicing](#_5hl5h3l7teel) with electronics, then we move into a few more detailed ones like [controllers and communication](#_toz8zwfuutx0), working with [inputs and outputs](#_1flrqkoezqgl), [data collection and visualization](#_prth69btljhq), and [arduino programming and debugging](#_hvp6e3mnwn51).
* Some software interfacing examples are provided in the [Appendix](#_ls8k9kf5h5nr) section, and I share a little about [my own projects](#_372qyuawcs69) for interest as well.
* I’ve also shared [my collection](#_4iaszhbsuo4i) of [MAKE magazine](https://makezine.com/) issues

*Supplemental files:* [*https://github.com/semoyerVT/electronicsGuide*](https://github.com/semoyerVT/electronicsGuide)

*Supplemental videos:* [*https://www.youtube.com/channel/UCbg7GRRLjmndbvHZ0qz7O6A*](https://www.youtube.com/channel/UCbg7GRRLjmndbvHZ0qz7O6A)

*Link to* [*this document*](https://docs.google.com/document/d/1QkrkIZ-MU8yB-zQCiiEAo70gQaDHsIA4mv2YUWBunoY)

[**Electronics Guide**](#_2v35fymqtr32) **1**

[Important Information](#_iq90bbgo85qj) 2

[Introduction](#_qmu41xgfqy3m) 2

[Shopping, Learning, and Practicing](#_5hl5h3l7teel) 3

[Controllers and Communication](#_toz8zwfuutx0) 4

[Controllers](#_gpk5b7o2joxl) 4

[Communication](#_6oxjl876bfkk) 5

[Inputs and Outputs](#_1flrqkoezqgl) 5

[Inputs](#_qsnfnv8xswol) 6

[Outputs](#_txgp9b3bf6ot) 6

[Data Collection and Visualization](#_prth69btljhq) 7

[Arduino Programming and Debugging](#_hvp6e3mnwn51) 8

[Programming](#_exbtge34ovam) 8

[Debugging](#_hoe020vwufaj) 9

[About Me and My Projects](#_372qyuawcs69) 9

[Some Definitions](#_edf6tabwl8ga) 10

[**Appendix**](#_ls8k9kf5h5nr) **11**

[MATLAB-Arduino Example](#_71zzu63cmh0k) 11

[WinForms Desktop App Example](#_gbnms7ipf1or) 12

[**MAKE: Magazine!**](#_4iaszhbsuo4i) **12**

[Important Note](#_ukqac3ax0qlh) 12

## Important Information

* **Open Source**: You are free to distribute and modify this document. If you have suggestions or comments for improvements and updates, please email them to me (semoyer@vt.edu), would be greatly appreciated!
* **RISKS**:
  + **Fire**: Yes, electronics can catch fire when mishandled. Be mindful of where you are doing this work, and work safely! Check out [this useful guide](https://www.chipwired.com/can-arduino-catch-fire/#:~:text=Using%20inferior%20quality%20components%20in,a%20fire%20to%20flare%20up.) to fire hazards associated with this type of work.
  + **Static Electricity**: Only you can prevent electrostatic discharge. I can’t tell you how many times I was careless in the past and fried a component, and only found out after spending hours debugging. Discharge your static before handling components, and don’t work with your socks on carpet. [Read this](https://www.dummies.com/programming/electronics/electronics-safety-lesson-guard-against-static-discharge/) for some more information, and [here](https://www.electroschematics.com/static-electricity-and-precautions/)’s a good guide for precautions.
  + The **Magic Blue Smoke**: If you wire something backwards (power to ground, ground to power, etc.), you run the risk of letting the blue smoke out of your MCU\* or component, and you can’t put that back in. If you set a pin up as an INPUT in your code, and you apply power to it, you can ruin that GPIO pin (or worse, your component or board). **Be very careful** how you set up your pins in code, and how you wire your components. There is usually an instructions page for any component you buy from Adafruit or Sparkfun, they’re usually very helpful.

*\*****MCU****: MicroController Unit, AKA controller, microprocessor, processor, board, dev board*

## Introduction

[INSERT VIDEO]

**Electronics**, or more specifically what I will call [Embedded Systems](https://www.omnisci.com/technical-glossary/embedded-systems) Design, is super cool. No matter what discipline you align with, it is a valuable skill to be able to design, build, and integrate electronics into your projects. **You might need** a simple data collection device for measuring moisture in soil, glucose in the body, carbon monoxide in the air, pressure on a surface, vibrations in a machine, proximity, or gyroscopic orientation of an object in space. The list goes on, and **you can do all this with some simple tools you teach yourself how to use!**

**Designing Embedded Systems** can seem daunting at first, but with practice you will be surprised just how quickly and easily you can build complex systems. On that same note, it is so very important that you KEEP IT SIMPLE as much as you can. Always start with the most basic functionality that accomplishes your goals. You can always improve/advance later, but I have learned too many times the hard way that **simplifying designs down the road is costly** (time, money, and especially sanity).

## Shopping, Learning, and Practicing

***How YOU navigate the World of Electronics!***

[INSERT VIDEO]

**Summary**

* Some great places to **Start**, to learn how to use components, and to **get inspired**!:
  + [Adafruit](https://www.adafruit.com/) and [Sparkfun](https://www.sparkfun.com/)
  + A few more worth mentioning: [Robotshop](https://www.robotshop.com/) and [DFRobot](https://www.dfrobot.com/) (carry a variety of brands), [Pololu](https://www.pololu.com/) (great for motors, motor controllers, and hardware) [others you might consider](https://www.circuito.io/blog/best-online-electronics-stores/)
* Places to shop (and compare prices) **once you know what you want** (large distributors, cheaper shipping): [Mouser](https://www.mouser.com/), [Digikey](https://www.digikey.com/), and [Arrow](https://www.arrow.com/)
* Places to **Learn**: These usually come up in a Google search when you have an idea or problem. Side note, become an expert Googler.
  + [Stack Overflow](https://stackoverflow.com/), [Code Project](https://www.codeproject.com/), [How To Mechatronics](https://howtomechatronics.com/), [Maker Advisor](https://makeradvisor.com/), [Instructables](https://www.instructables.com/), [Hackaday](https://hackaday.io/), [Arduino Tutorials](https://www.arduino.cc/en/Tutorial/HomePage), [Circuit Basics](https://www.circuitbasics.com/), [Tutorials Point](https://www.tutorialspoint.com/), [All About Circuits](https://www.allaboutcircuits.com/), [Stack Exchange](https://electronics.stackexchange.com/), and you can even find answers on places like Reddit.
  + Specific to learning programming languages: [Tutorials Point](https://www.tutorialspoint.com/), [W3 Schools](https://www.w3schools.com/)
* Places to **Practice** design and programming: [TinkerCAD](https://www.tinkercad.com/), [Fritzing](https://fritzing.org/), [Arduino Create](https://www.arduino.cc/en/Main/Create), [CodeBender](https://codebender.cc/), and there’s more out there these days I’m sure. I like to think of these more as a way to communicate designs, and practice really comes from just trying things out (ideally with a lot of research supporting decisions).
* Typical **Prototyping** components: [Hook up wire](https://www.adafruit.com/product/153), [breadboards](https://www.sparkfun.com/products/12002), and a power supply (see Power Tip in the [Inputs and Outputs](#_1flrqkoezqgl) section). Some [alligator clips](https://www.adafruit.com/product/1008) can be really handy too.
  + You might consider getting [a small kit](https://www.sparkfun.com/products/13973) of ‘discrete components’ that includes resistors, capacitors, diodes, LEDs, voltage regulators, etc. If you stick to basic MCU boards and breakout sensors, you likely **do not need** these types of components, but they’re great to have on hand.
* **Tools** you might need:
  + [**Digital multimeter**](https://www.sparkfun.com/products/12966) (you can get them really cheap from places like [Harbor Freight](https://www.harborfreight.com/7-function-digital-multimeter-63759.html)). If you are working with electronics, you really should [know your way around a multimeter](https://learn.adafruit.com/multimeters).
  + Simple multi-tool (like the kind [you see](https://images-na.ssl-images-amazon.com/images/I/81pWR1LEO9L._AC_UL1500_.jpg) at a gas station) or just cheap **pliers** will do.
  + Simple set of **small screwdrivers** (or a [classic pocket combo](https://www.parallax.com/product/parallax-pocket-screwdriver/), my preference).
  + I like having a [cheap rubber adjustable clamp](https://www.harborfreight.com/hand-tools/clamps-vises/12-in-ratcheting-bar-clampspreader-62123.html) on hand as well (hold things still while you are soldering, attaching components, etc.).
  + Typically you can get away with **using things ‘around the house’** like nail clippers (trimming leads on breakout boards), tweezers (to grab the little things), a letter opener (to lift components off breadboards), clothes pins (hold things together), and hand sanitizer (to dry the oils on your hands when handling components).
  + If you intend to [solder components](https://learn.adafruit.com/how-to-solder-headers?view=all) at home, you’ll need a [**soldering iron**](https://www.robotshop.com/en/elenco-sr-2b-deluxe-25w-soldering-iron.html) and [solder](https://www.robotshop.com/en/lead-free-solder-100g.html), and likely [wire](https://www.adafruit.com/product/290) and [wire strippers](https://www.robotshop.com/en/elenco-st-30-wire-stripper.html). See PrototypingTip below.
  + **Patience**: This is a powerful tool. Navigating the world of electronics takes time, and it will be full of challenges to overcome. Sometimes 90% of the battle is researching to find the right component for the job; lean on sites mentioned in the [shopping section](#_5hl5h3l7teel).

***Shopping Tip:*** *Shop around for the same and similar parts once you know what you want, and also try to balance* ***using as few vendors as possible****. It'll go a lot smoother if your BOM (bill of materials, what you need to buy) has all electronic components available from 1 or 2 vendors instead of 5, and that the vendors are well known and established (and not Ebay, not Amazon, and not Alibaba).*

***Prototyping Tip:*** *You will very likely need to do some soldering, as many microcontrollers, sensors, and other components do not come with their breadboard headers pre-soldered. Keep this in mind, as* ***you will want someone experienced to assemble the components****. See note about Frith Lab in the* [*important info*](#_iq90bbgo85qj)*.You might be able to find everything you need already fully assembled (look for ‘dev kits’), but they usually cost more. Check out the notes in the* [*controllers section*](#_gpk5b7o2joxl) *for places to shop for dev kits (components that connect directly without the need for soldering).*

## Controllers and Communication

***How embedded systems Think and Talk!***

[INSERT VIDEO]

### Controllers

* The [**Arduino**](https://www.arduino.cc/en/guide/introduction)platform is powerful, simple, and well supported. Most popular microcontroller boards (MCU’s) are supported by Arduino, so you don’t have to get an Arduino-specific MCU! Not to downplay the impressive capabilities of other platforms like [RasPi](https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/), [BeagleBone](https://beagleboard.org/black), and whatever Intel is up to these days, but for simplicity and function Arduino is hard to beat.
* Picking **the ‘right’ MCU** can be confusing, so here are a few suggested Arduino-compatible boards that are cheap and effective for most applications
  + [Arduino Uno](https://www.sparkfun.com/products/11224) (the classic go-to board, still awesome)
  + [Arduino Pro Mini 5V](https://www.adafruit.com/product/2378), also comes in 3.3V (see power tip below)
  + [Adafruit Huzzah](https://www.adafruit.com/product/2471) (WiFi capable)
  + [Adafruit Trinket 5V](https://www.adafruit.com/product/1501) (definition of cheap and effective, also has a 3.3V version)
  + [Sparkfun Thing](https://www.sparkfun.com/products/13231) (WiFi plus an on-board LiPo battery connection and charger!)
  + [Teensy](https://www.sparkfun.com/products/15583) (one of my personal favorites, a little ridiculous how much this thing can do, way more than you need, great for sound output)
  + [Seeed Xiao](https://www.seeedstudio.com/Seeeduino-XIAO-Arduino-Microcontroller-SAMD21-Cortex-M0+-p-4426.html) (low cost and powerful, new to the market)
* Entire **Dev platforms** you might consider (not usually my style, but you might like):
  + [Adafruit Feather](https://www.adafruit.com/feather) (awesome stackables)
  + [Seeed-Studio](https://www.seeedstudio.com/) (Grove devices are popular, I’m not the biggest fan though)
  + [Adafruit Flora](https://www.adafruit.com/index.php?main_page=category&cPath=92) (perfect for wearables!, lots of [tutorials](https://learn.adafruit.com/category/flora) available as well)
  + [Tiny Circuits](https://tinycircuits.com/pages/tinyduino-overview) (they’re cute, easy to use)
  + [ExpressIf](https://www.espressif.com/en/products/socs/esp32) (fancy wireless tech, but surprisingly affordable)

### Communication

* **Internal** (within the embedded system):
  + [Serial](https://www.ladyada.net/learn/arduino/lesson4.html): Many MCUs have multiple serial ports available, where one is dedicated to programming with USB. Some sensors and other components use regular old serial for communication. Just remember that RX on device 1 goes to TX on device 2, and vice versa.
  + [SPI](https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all) and [I2C](https://learn.sparkfun.com/tutorials/i2c/all): These are the common internal interfaces you will find, and usually with sensors. Don’t be scared, just follow the example code and **hook things up like they show you in tutorials and you’ll be fine**. Just remember I2C connections on both ends (device to device) match (SDA to SDA, SCL to SCL), SPI connections match on both ends except MISO and MOSI which are crossed between the two devices.
* **External** (interfacing with the embedded system):
  + USB Serial: For programming your board, and also for communication with a terminal (like Arduino’s [serial monitor](https://learn.adafruit.com/adafruit-arduino-lesson-5-the-serial-monitor/the-serial-monitor)) or other applications (custom desktop app, Putty, etc.).
  + **WiFi**: The best approach nowadays is to buy MCUs with WiFi built in (see Controllers section). But for those that need to add WiFi to something like an Uno, here’s a good ol’ [ESP8266](https://www.sparkfun.com/products/17146) breakout.
  + **Bluetooth**: The [Smirf series](https://www.sparkfun.com/products/12577) of BLE modules are kind of the gold standard for robust bluetooth. But, a class HC-series works just fine for most applications, here’s a [tutorial](https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/).
  + So many [more wireless](https://www.sparkfun.com/pages/wireless_guide): [XBee](https://www.sparkfun.com/pages/xbee_guide) ([tutorial](https://www.instructables.com/How-to-Use-XBee-Modules-As-Transmitter-Receiver-Ar/)), General RF, Cellular (where the IoT came from!), GPS, etc. There are many ways you can even get creative with how devices communicate (what about light, sound, touch, speech, biometric feedback).

***Power Tip:*** *Determine if you want to design* ***a 5V system or a 3.3V system*** *(or ‘l*[*ogic level*](https://learn.sparkfun.com/tutorials/logic-levels/all)*’). You will find that many sensors only support one or the other. You can use* [*level shifters*](https://www.adafruit.com/category/864)*, but that can become a pain quick. Most MCU’s that are labeled as 5V usually support 3.3V sensors, but not vice versa.*

***IoT Tip****: What is the* ***Internet of Things****? Really it’s an idea that with the advances in the interconnectivity of all ‘things’ we sense and act on with electronics, there’s ways to collect, display, and analyze data collected from devices (as well as control the devices) using wireless technology. Oh, and way cheaper than we ever knew prior to this* [*fourth industrial revolution*](https://www.zdnet.com/article/how-iot-will-drive-the-fourth-industrial-revolution/)*, industry 4.0, maker movement, or whatever you want to call it that has been gaining momentum in the past 10-15 years.*

## Inputs and Outputs

***How embedded systems Sense and Act!***

[INSERT VIDEO]

### Inputs

* **Sensors**! Arguably **the most important parts of your embedded system**. I was taught to always ‘spend the money on the sensors,’ which makes sense because if you don’t have reliable measurements, nothing else downstream is going to be reliable either.
  + [Touch](https://www.adafruit.com/product/1374), [temperature](https://www.adafruit.com/product/386) and humidity, [proximity](https://www.adafruit.com/product/2168), [motion](https://www.adafruit.com/product/189), [distance](https://www.adafruit.com/product/4007), [sound](https://www.sparkfun.com/products/12642), light [intensity](https://www.adafruit.com/product/439) or [reflectance](https://www.pololu.com/product/4101), [color](https://www.adafruit.com/product/1334), [weight](https://www.adafruit.com/product/4630) and [pressure](https://www.adafruit.com/product/1893), [vision](https://www.adafruit.com/product/1906), and so much more.
  + From a **biological perspective**, think about all the ways that you (and other organisms) ‘sense’ the world around you, interact with it, and [connect that](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3274279/) with the type of ‘sensing’ that you are hoping to accomplish in a project.
* **Other Inputs**:
  + [Buttons](https://www.adafruit.com/product/1400), [keypads](https://www.adafruit.com/product/3844), knobs (potentiometers, [linear](https://www.adafruit.com/product/4219) and [rotary](https://www.adafruit.com/product/4133)), [touch screens](https://www.adafruit.com/product/1770), data [stored](https://www.adafruit.com/product/254) or transferred into your system, and more.
  + **Biometric** feedback: [Muscle contraction](https://www.adafruit.com/product/2699) (EMG), [heart rate](https://www.sparkfun.com/products/12650) (ECG), g[alvanic skin response](https://www.seeedstudio.com/Grove-GSR-sensor-p-1614.html) (GSR, “sweating”), glucose monitoring, etc. Check out Seeed studio’s [collection](https://wiki.seeedstudio.com/Sensor_biomedicine/), and other major vendors have their own too.
  + **Settings and configuration**: You might have variables in your code, or otherwise brought into your system that are used to adjust things. These are inputs too. Maybe you have a file with calibration data that’s used to set up your arduino code, or a flag (bit, boolean, etc.) that you set for testing.

### Outputs

* **Motion** (aka locomotion, movement):The basic building blocks of motion are **framing** (supporting structures), **actuators** (motors, power transmission such as gears, anything converting energy/doing work), and **control** (the system manipulating the motion).
  + From a **biological perspective**, you can think of frames like [skeletons](https://www.dummies.com/education/science/biology/types-skeletal-systems/), actuators like [muscles](https://www.khanacademy.org/science/high-school-biology/hs-human-body-systems/hs-the-musculoskeletal-system/v/three-types-of-muscle), and control like [nervous systems](https://organismalbio.biosci.gatech.edu/chemical-and-electrical-signals/nervous-systems/).
* **Motors**! This is where things can get a little hairy, since **you care about power** requirements (see power tip). Small servo-motors, haptic devices, and DC motors may not need an ‘external power supply’ if they are low voltage and low current. Know what the limitations of your MCU are in terms of voltage (5V or 3.3V) and current supply (such as through the USB cable, which isn’t much).
  + Common [servo-motors](https://www.adafruit.com/product/169), [stepper motors](https://www.adafruit.com/product/324) (more complex), and [DC brushed motors](https://www.adafruit.com/product/711)
  + Common [haptic](https://www.adafruit.com/product/1201) devices (vibration) and [solenoids](https://www.adafruit.com/product/3992) (push and pull, like a latch)
* **Other outputs**:
  + Light ([LED](https://www.adafruit.com/category/37)) and Display ([OLED](https://www.adafruit.com/product/938), [Character](https://www.adafruit.com/product/181), [Digit](https://www.adafruit.com/product/1907)), even a few [terminals](https://www.seeedstudio.com/Wio-Terminal-p-4509.html) out there now, [some](https://www.seeedstudio.com/ESP32-Development-board-WT32-SC01-p-4735.html) built right on top of ESP32 tech for easy wireless interfacing
  + Sound: [Speakers](https://www.adafruit.com/product/3968) often require [additional components](https://www.sparkfun.com/products/11044) and [external power supply](https://learn.adafruit.com/power-supplies), though some are small and simple enough to be used directly with a MCU (such as a [piezo buzzer](https://www.adafruit.com/product/160) which you can [make music](https://www.arduino.cc/en/Tutorial/BuiltInExamples/toneMelody) with!)
  + Vibration: [Surface transducers](https://cdn-shop.adafruit.com/1200x900/1674-00.jpg), [motors](https://www.robotshop.com/en/seeedstudio-mini-vibrating-motor.html), and other haptic devices
  + [Fans](https://www.sparkfun.com/categories/tags/fan): Might require external power supply depending on the size
  + [Pumps](https://www.adafruit.com/product/4547) and [Vacuums](https://www.adafruit.com/product/4700): It’s incredible how small you can find these. However, I strongly advise that you **do NOT use pneumatics** or work with fluid pressure in any way for your projects. It’s dangerous, and can become very complex quickly.
  + Data! Yes, data is an output when you are collecting it, displaying, storing, etc.

***Power Tip:*** *Some outputs demand too much power (higher voltage and/or higher current demand) than the MCU board can provide through a USB cable. You might need to get a battery holder, rechargeable pack, or a wall adapter to use as an external*[***power supply***](https://learn.adafruit.com/power-supplies)*.*

* *For example, if you have a simple 6V DC motor that draws 0.5A of current when loaded, you can probably get away with just using a 4xAA battery pack (get a couple* [*cheap holders*](https://www.robotshop.com/en/battery-holder---4-x-aa-compact.html) *and some AA batteries). Use that to supply power directly to whatever* [*motor driver*](https://www.adafruit.com/product/3190) *you use, and optionally to power your MCU (using the voltage input pin).*
* *If you’re using 5V outputs, a simple* [*5V wall adapter*](https://www.adafruit.com/product/276) *might be all you need. You can find them for* [*12V systems*](https://www.adafruit.com/product/798) *too, but be mindful of the MAX input voltage your MCU can handle.*

***Build Tip:*** *As you think about what to build your devices out of, consider checking out places like* [*Vex Robotics*](https://www.vexrobotics.com/) *and* [*Parallax*](https://www.parallax.com/)*, or any other where you can get kits to build things (compatible motors and hardware). It could* ***make your lives way easier*** *than custom-building everything. A good engineer knows when to build, and when to buy. Think about time, cost, effort, and what your end goals are (remember, you’re not trying to sell your devices).*

## Data Collection and Visualization

***How embedded systems Collect and Display!***

[INSERT VIDEO]

**Summary**

* On-Board **Storage**: There is ‘static’ memory (doesn’t get overwritten when you program the device) available on most MCU’s, should you need it. The common ones are [EEPROM](https://www.arduino.cc/en/Reference/EEPROM) (easiest) and [SRAM](https://learn.adafruit.com/memories-of-an-arduino/arduino-memories). You can also add [SD card](https://www.adafruit.com/product/254) holders to your system.
* **MATLAB**: My opinion so far is that if you are doing something fairly simple like taking a measurement on a single analog pin (reading a simple sensor) with a standard MCU like an Arduino Uno, [MATLAB-duino](https://www.mathworks.com/hardware-support/arduino-matlab.html) is a great option. What I don’t like is that as things become more complex, or non supported boards (see [supported hardware](https://www.mathworks.com/hardware-support/arduino-matlab.html)), MATLAB can become very challenging.
  + An [example](#_71zzu63cmh0k) is provided in the appendix!
* **Desktop Applications**: There are many options for creating [your own applications](https://docs.microsoft.com/en-us/windows/apps/desktop/) to manage and interface with your embedded systems. The potentials are virtually limitless, but here are some common uses:
  + Stream live data from your Arduino to a [WinForms](https://docs.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio), [Console](https://docs.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-console), or WPF app
  + Store data to [Excel sheets](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/interop/how-to-access-office-onterop-objects) for data analysis
  + Interface with external applications using their APIs (such as streaming data straight into your Google sheets, docs, etc. using their [Cloud Platform](https://console.developers.google.com/)).
  + Make simple real-time charts with [.NET charting](https://www.i-programmer.info/programming/uiux/2756-getting-started-with-net-charts.html) (there are other free ones).
* **Browser-Based** Data Streams:
  + Stream to a webpage on your PC! [Here](https://circuitdigest.com/microcontroller-projects/sending-arduino-data-to-webpage)’s an example, and [here](https://www.circuitbasics.com/how-to-set-up-a-web-server-using-arduino-and-esp8266-01/)’s another, there’s tons of these out there. The WiFi device acts as a modem, serving up a webpage, and you connect to it’s network with your PC.
  + [Plot.ly](https://plotly.com/): There’s others like this out there, but I’ve only used this one. [Here](https://hackaday.io/project/5626-plotly-arduino-esp8266)’s an example, and [another](https://www.instructables.com/Plotly-Arduino-Data-Visualization/).
* **SmartPhone** Applications:
  + [MATLAB Mobile](https://www.mathworks.com/products/matlab-mobile.html): Interface MATLAB with your mobile devices!
  + Custom [App Development](https://visualstudio.microsoft.com/vs/features/mobile-app-development/) with Visual Studio! Recommend cross-platform platforms and languages such as [Xamarin](https://docs.microsoft.com/en-us/visualstudio/cross-platform/cross-platform-mobile-development-in-visual-studio?view=vs-2019), but there are others. There are even [emulators](https://visualstudio.microsoft.com/vs/msft-android-emulator/) available for testing!

***Developer Tip****:* [*Visual Studio*](https://visualstudio.microsoft.com/) *is an incredibly powerful platform, and the Community edition is FREE for you to use on your Windows or Mac PC. Develop desktop apps, cross-platform apps, browser apps, mobile apps, game development, and so much more. There are often options for the language(s) used to develop, though for you all I would recommend* [*C#*](https://www.w3schools.com/cs/) *or* [*Visual Basic*](https://www.vbtutor.net/lesson1.html)*.*

## Arduino Programming and Debugging

***How embedded systems Know what to do!***

[INSERT VIDEO]

### Programming

* Install the [Arduino IDE](https://www.arduino.cc/en/software) application
* First step, **Blink**:
  + Open *File -> Examples -> Basics -> Blink*: You may need to replace ‘LED\_BUILTIN’ with the specific pin that your MCU has an on-board LED (or add an external LED to any pin).
  + Using the Tools menu, choose your Arduino-compatible board and COM port. If your board is not listed, add it using the [Boards Manager](https://support.arduino.cc/hc/en-us/articles/360016119519-How-to-add-boards-in-the-board-manager), and if needed add a URL to an external board (such as [Sparkfun](https://learn.sparkfun.com/tutorials/installing-arduino-ide/board-add-ons-with-arduino-board-manager) boards).
  + Upload Code, observe the blinking LED on your MCU!
* Also a first step, **Hello World**:
  + Open *File -> Examples -> Communication -> ASCII Table*: Follow steps above for setup (selecting board and COM port). When you upload you should be able to open the Serial Monitor and see a full ASCII table printed out.
* Structure of Arduino code (and usually **in this order**):
  + **#include**: For including libraries in your code. Some are internal (native to Arduino) and external (such as is provided by the component manufacturer, and usually available on Github, [here](https://www.arduino.cc/en/guide/libraries)’s a little more about those, and [another](https://www.instructables.com/How-to-Add-an-External-Library-to-Arduino/)).
  + **#define**: Use [these](https://www.arduino.cc/reference/en/language/structure/further-syntax/define/) sparingly, but they can be really handy. When you have something like ‘*#define DUMMY\_PRINT Serial.println(‘dummy’)*’... anywhere in your code where you put ‘*DUMMY\_PRINT;*’ it will REPLACE it with ‘*Serial.println(‘dummy’)*’ right before the code compiles (which happens automatically when you click Upload).
  + **Global** variables: Variables (int, double, bool, string, etc.) that are declared ([scoped](https://www.arduino.cc/reference/en/language/variables/variable-scope-qualifiers/scope/)) **outside of the functions** (and usually at the top of the code), that are shared throughout the whole code file.
  + **setup()**: This function runs first, once, and right before the loop().
  + **loop()**: This function runs after the setup(), and repeats until MCU is powered off or reset.
  + **Custom** functions: You can make your own functions, check out this [tutorial](https://www.tutorialspoint.com/arduino/arduino_functions.htm).
  + **Local** variables: Variables are declared and used **inside functions** (setup, loop, and custom) and are only accessible within that function.

### Debugging

* You will likely spend just as much time (if not a lot more) debugging your embedded system (code, wiring, etc.) than actually writing code and hooking things up. It takes practice, but here are some good strategies to finding ‘the bug’.
  + Check your **wiring**, every wire.
  + Check your **‘pin mapping’**, if your sensor pin is A2, does your code use A2?
  + Add your own **custom 'checkpoints’**
    - With USB connected, using the Arduino IDE Serial Monitor, add things like ‘Serial.println(“LED should be on”)’ and ‘Serial.println(“Here”)’ after various places in your code, make sure the execution gets to those points and your peripherals (inputs and outputs) act as you expect.
  + Replace your sensor data with **dummy data** (comment out code that generates data, replacing the output variable with a dummy/example value). The idea here is to eliminate the sensor as a culprit, or to verify that it IS the culprit.
  + There are lots of strategies to track down bugs, [here is a good article](https://www.circuito.io/blog/arduino-debugging/) about this.
* [Visual Micro](https://www.visualmicro.com/): For those more adventurous, consider adding the Visual Micro package to your Visual Studio app. It can do everything the Arduino IDE and a lot more, including breakpoints!

***Programming Tip****: Buy components that have clear examples and tutorials! I love Adafruit for this reason, but you can also find nearly anything through some savvy Googling (see notes in the* [*shopping section*](#_5hl5h3l7teel)*). You will become a PRO in reverse engineering example code, so that you can then integrate a component into your custom code. Run the examples first, to make sure it works as expected.*

## About My Projects

[INSERT VIDEO]

**Summary**

* My name is **Stephen Moyer** and I work as a GTA for the first year engineering program at Virginia Tech. I am currently pursuing a PhD in Engineering Education. Previously I have worked in the aerospace and automotive industries, in roles from design and manufacturing. My expertise is in embedded systems design and software development, though I’ve also worked with controls, systems, and mechanics. My BSE is in Mechatronics from UNC Asheville and NC State.
* A Maker or a Roboticist? Or a Robotics Engineer? What’s the Difference!?
  + **Being a Maker** is more of an idea than a title, it crosses discipline lines and transcends educational backgrounds. Leading engineers in industry might identify as a Maker, but so might an elementary school student learning STEM. Being a Maker is about creativity, sharing, and excitement around learning. It is not just for those that build electronics, but for anyone that creates and shares!
  + **Being a Roboticist** is another story, though one might argue that anyone that builds robots is a roboticist. I say, as long as you're learning and enjoying the experience working with robotics, you’re a Roboticist too!
  + **Being a Robotics Engineer** is perhaps a little more straightforward, which is an engineer that works with robotics. You can work in a ton of different fields (science, medicine, agricultural, etc.) and work with robotics in many different ways, and thus identify as a ‘Robotics Engineer’. Any electro-mechanical device can be considered a robot. **Mechatronics** is an engineering field that specifically focuses on the cross-roads of mechanical, electrical, and computer theory, and is often associated with robotics.
* **Undergraduate** Electronics Projects:
  + Major Capstone: **Interactive Mechanical Mirror**
    - Arduino + Vision + Motors + WPF(C#)
    - Desktop app with an XBox Kinect sensor, driving a matrix of servo motors over USB communication with an Arduino-based motor controlling system
  + Minor Capstone: **Feedback Control Educational Platform**
    - Arduino + Reflectance Sensors + Motorized Potentiometers
    - An educational tool for the manipulation and visualization of control algorithms in real time for a line tracking device, using motorized potentiometers with mounted reflectance sensors and embedded on the Arduino platform.
  + SoutheastCon Competitions: **Robots**! (my favorite projects)
    - A variety of technologies used in developing these robots over several years.
* **Industry** Electronics Projects:
  + Altec Industries: Manufacturing Engineer
    - **Oven Temperature Monitor System**
      * Arduino + Thermocouples + SQL + WinForms (VB)
      * Thermocouples installed into the sides of a large curing oven for the real time visualization of heat distribution, using Arduino-based platform for sensing and SQL for data storage. Desktop applications for the visualization and analysis of data.
    - **E-Coat Anode Current Monitor System**
      * CPP + Current Transducers + WiFi + Node.JS + SQL + WinForms
      * Current transducers (passive DC current measurement) mounted on high voltage cables for the visualization of current distribution of anode towers within an electrolytic coating system, using ARM-based platform for sensing and SQL for data storage. Desktop applications for the visualization and analysis of data.
  + Boeing Research and Technology: Software Engineer
    - **Remote Optical Control Surface Indication**
      * CPP + XBee + Python + Vision + Lasers + WPF(C#)
      * Patented technology leveraging laser line diodes, linear CCD array vision, and aircraft kinematics and geometries to determine deflection of moving control surfaces in real time.
* Please do reach out (email me) **if you have any questions**, happy to help. You will find most common answers in this document, so be sure to have a good read through.

# Appendix

## MATLAB-Arduino Example

Also available to Download: <https://github.com/semoyerVT/electronicsGuide/blob/main/MATLAB-Arduino.docx>

**Instructions for setting up Arduino Support for MATLAB:**

**Prerequisites**: Windows OS with MATLAB and [Arduino](https://www.arduino.cc/en/software)\* installed

\* DO NOT install Arduino using Windows App Store, just download the regular Windows installer (or ZIP). There are still bugs with using the Windows Store, after all these years launching Windows 10.

**NOTE**: This method does not allow you to directly modify the Arduino code, MATLAB handles that for you and you write code in MATLAB to control pins and interface with peripheral devices

1. Go to <https://www.mathworks.com/hardware-support/arduino-matlab.html>

2. Click 'Get support package' which will download the package

3. Run the executable, which will open your Add-On Manager for MATLAB (it may prompt you to sign in)

4. Accept license agreement and click 'Next' to download and install the Arduino third-party package

5. Follow prompts for installer, which may require giving app permissions

6. When complete, click 'Setup Now' or type '**arduinosetup**' in the command window

7. To setup the Arduino-MATLAB connection, you will have to select your board ('uno' for an Arduino Uno) and port (i.e. 'COM5')

The default communication libraries are I2C and SPI, and most sensors you work with will use one of those protocols

Click 'Program' to write the MATLAB Arduino Server to your board, which will allow you to use MATLAB to control

8. From the command window, create an instance of your board (i.e. **myBoard = arduino('com5','uno')**)

9. From the command window, TEST your connection by turning the built-in LED on and off (i.e. **writeDigitalPin(myBoard,'D13',true)**)

Verify that the LED on the board turns on, then turn it back off (i.e. **writeDigitalPin(myBoard,'D13',false)**)

10. You're good to go, check out the '[MATLAB Support Package for Arduino Hardware](https://www.mathworks.com/matlabcentral/fileexchange/47522-matlab-support-package-for-arduino-hardware)' help documentation for examples etc.

11. For reading an analog pin (such as with a temperature sensor), you can use **readVoltage(myBoard,'A1')** and then do the math on the MATLAB side

## WinForms Desktop App Example

Also available to Download:

<https://github.com/semoyerVT/electronicsGuide/blob/main/WinForms-Arduino.docx>

Video Walkthrough: <https://youtu.be/OYlVoxesuv0>

**Instructions for creating a desktop application to store Arduino data in Excel files:**

**Prerequisites**: Windows OS with [Visual Studio Community](https://visualstudio.microsoft.com/vs/community/) installed (make sure you have ‘desktop development’ selected in the installer), and [Arduino](https://www.arduino.cc/en/software)\* installed (optionally Visual Micro, see [debugging](#_hoe020vwufaj) section).

* If you don’t want to view/modify code, **you don’t really need to install Visual Studio**. After unzipping from step 1 below, you can just run the EXE directly which is located at B\_B\_C\_A -> bin -> Debug.

1. Go to <https://github.com/semoyerVT/electronicsGuide> and **download** the ‘Bare Bones’ folders, one has the full Visual Studio solution, and the other the Arduino code (‘MCU’).

2. Once **unzipped**, double click the solution file (.sln) in the App folder and it should open up in Visual Studio (or go to Visual Studio, and open that solution that way).

3. **Upload the Arduino code** to your compatible board (don’t forget to set your board and COM port in the Tools menu).

5. **Connect USB** to the PC and your Arduino. For a quick test, open the Arduino IDE Serial Monitor to see the data stream (it’s just random data every 2 seconds).

6. **Start the App** in Visual Studio (defaults to Debug mode, which is fine). Follow instructions shown on the window for use (pick COM port, open port, optionally set file path and log to Excel).

# MAKE: Magazine!

These are FULL of inspiration and knowledge, and cover literally every corner of the Maker electronics world over the last 16 years. Quite an incredible story of this ‘Maker revolution’ is told through these magazine issues. **Please enjoy!**

**Disclaimer**: I’ve set up permissions so that you can view these magazine issues, but you cannot download, print, copy, share, or otherwise distribute in any way. This is to protect the rights of the magazine while also giving you access to enjoy the content.

*You can find summaries of every issue* [*here*](https://www.makershed.com/collections/make-magazine)*. I don’t know if there is a better search engine out there for finding specific things in these issues, let me know if you come across one!*

|  |  |  |  |
| --- | --- | --- | --- |
| [Volume 1: Feb 2005](https://drive.google.com/file/d/10CuLmBslwf6aGl1xRBWLLmHi8e94lVTV/view?usp=sharing) | [Volume 21](https://drive.google.com/file/d/1FC2KbnL7h4BV7MaV5DApxvBponimaFI9/view?usp=sharing) | [Volume 41](https://drive.google.com/file/d/1nsxHsaCL_aAAdk8gBAizcrzSBmM1IeYt/view?usp=sharing) | [Volume 61](https://drive.google.com/file/d/1XbZl7z-ZBRG5-0aLErbu7W9lRxwlFtun/view?usp=sharing) |
| [Volume 2](https://drive.google.com/file/d/1HEQPHsYbE5ETQQQmP7Hh9x8beKxCKGly/view?usp=sharing) | [Volume 22](https://drive.google.com/file/d/1Gwciyuk5zL2BkeTejHk0iTlrAHoW-4aX/view?usp=sharing) | [Volume 42](https://drive.google.com/file/d/1Mq2o1KW2HcpC-swOv9s_mJLN1D_7UKX5/view?usp=sharing) | [Volume 62](https://drive.google.com/file/d/1xdgvL0Z9z15NCg874foXjCBKrJkqnT52/view?usp=sharing) |
| [Volume 3](https://drive.google.com/file/d/1BHGCIyxfmWyYCvJLT4xopiWPr42bbaZa/view?usp=sharing) | [Volume 23](https://drive.google.com/file/d/1XqRi7-3ssbTatrYQiCOda7aSVzhxjzl5/view?usp=sharing) | [Volume 43](https://drive.google.com/file/d/1tZLp09-UemwXaIGO9mUMvUSAfIa7SFCK/view?usp=sharing) | [Volume 63](https://drive.google.com/file/d/1x7DtRgoEq1E2IB4k_11MAD1LGzbbRTG7/view?usp=sharing) |
| [Volume 4](https://drive.google.com/file/d/1QKpKN3nAC03Kcl4e7vf1k86iPL_fS29l/view?usp=sharing) | [Volume 24](https://drive.google.com/file/d/1oX6z4BMBlMtpfAu1ikUX4nKpW4SgX8mk/view?usp=sharing) | [Volume 44](https://drive.google.com/file/d/1YMZck6LtNEkeSFNm5LqwC7LhJscSV93_/view?usp=sharing) | [Volume 64](https://drive.google.com/file/d/1VQ7clN9V5S5gTzmh1F-O55p8EGk9SZFA/view?usp=sharing) |
| [Volume 5](https://drive.google.com/file/d/1-Wbf_TiNrLGY9ofaDPUTWU8GdjKJfdTx/view?usp=sharing) | [Volume 25](https://drive.google.com/file/d/1DDdO4KlO62mQBLuOu8y2mD3ItdWtL4KL/view?usp=sharing) | [Volume 45](https://drive.google.com/file/d/1mQv6hEGGD2CutFhW98HcLmFsPaqkTFeY/view?usp=sharing) | [Volume 65](https://drive.google.com/file/d/1gpopsSplUSz6WWSpTzi5R9fpXi6DxQNw/view?usp=sharing) |
| [Volume 6](https://drive.google.com/file/d/1rVCl6KvMPBuH8f5ho_HTsBriwl5wamtC/view?usp=sharing) | [Volume 26](https://drive.google.com/file/d/1TMc30hp-M7dVDBI6ri-Hl-Vms87dI7HE/view?usp=sharing) | [Volume 46](https://drive.google.com/file/d/14lY2cs6LW8emZmH66AMGeZFsbp81TK25/view?usp=sharing) | [Volume 66](https://drive.google.com/file/d/1dicMOsUReFsZ9seU_OZ0boWbaRlw20bg/view?usp=sharing) |
| [Volume 7](https://drive.google.com/file/d/1bJN_Jk6Qqt0h3romtdDIFKIBByqZkEc4/view?usp=sharing) | [Volume 27](https://drive.google.com/file/d/1prR_9tSFCOUjHg38eC31b4xwKZ6hh-Bq/view?usp=sharing) | [Volume 47](https://drive.google.com/file/d/149whiTLp1d4hxDdt49hObV4Exyr4xld_/view?usp=sharing) | [Volume 67](https://drive.google.com/file/d/1kWo8YHCv62thWqfINfjvKBb8ygVEqTaT/view?usp=sharing) |
| [Volume 8](https://drive.google.com/file/d/1jnbGl-pSC0VeYWodAzzrqMbTdRexvbZG/view?usp=sharing) | [Volume 28](https://drive.google.com/file/d/1yQVuJXW7HtCVlvMDA4yvFALN1CVpUE7C/view?usp=sharing) | [Volume 48](https://drive.google.com/file/d/1jDV-SxOPsHGmNi7-RpdOjrnSLww46eGc/view?usp=sharing) | [Volume 68](https://drive.google.com/file/d/1qr-aSXfZSIq6p0h9EBnP2o6ae-LWriPk/view?usp=sharing) |
| [Volume 9](https://drive.google.com/file/d/1WCihtxYqvxVY0prIh29eQFbjFNlqU3NX/view?usp=sharing) | [Volume 29](https://drive.google.com/file/d/1UKJjoquQ1RK124fRAkCfDb_prZoLHnjz/view?usp=sharing) | [Volume 49](https://drive.google.com/file/d/11XU74CIO0I7qx-5UUQBlvJzsHebKk1q0/view?usp=sharing) | [Volume 69](https://drive.google.com/file/d/168FzrLMNAt5e74iTtJu5y7gSi6_oblEn/view?usp=sharing) |
| [Volume 10](https://drive.google.com/file/d/1EaXLQC-YsBXBRrSXblayW798yPGB1uex/view?usp=sharing) | [Volume 30](https://drive.google.com/file/d/1ymvgFtXUpLxAX73T92M70NZ_MBc6hkwk/view?usp=sharing) | [Volume 50](https://drive.google.com/file/d/1z7SZL0r6-AKAgYCkAH6mCcWqMi265eu4/view?usp=sharing) | [Volume 70](https://drive.google.com/file/d/1IXR_T-5j_CToNqpgeaFv3QDRWRcY4zGG/view?usp=sharing) |
| [Volume 11](https://drive.google.com/file/d/1zzZcA2POf1tISr_WuZaUKydot4nFZu6n/view?usp=sharing) | [Volume 31](https://drive.google.com/file/d/1PAsDvSwWbNk5bm0Y0DhBIrLtMVcd80Q4/view?usp=sharing) | [Volume 51](https://drive.google.com/file/d/1H_a4IHFwdNhSPOm7mwOmZpgZ4UTRUYOL/view?usp=sharing) | [Volume 71](https://drive.google.com/file/d/1veU0r2P7e3KAOpkvVaSbce-T6LzyC18z/view?usp=sharing) |
| [Volume 12](https://drive.google.com/file/d/1O3yot1HkkFtw1iZA96UTZhIbFkqKpryC/view?usp=sharing) | [Volume 32](https://drive.google.com/file/d/1K9MwNZU4CMBQKY1dnh070XsAAXq8KTlP/view?usp=sharing) | [Volume 52](https://drive.google.com/file/d/1OmV9K_c9o1CS7_ZsbNDdi6qpy8MQu8_Z/view?usp=sharing) | [Volume 72](https://drive.google.com/file/d/1kVyY57bGUGKK6A2DM9jNFRiyEvxG7-Lu/view?usp=sharing) |
| [Volume 13](https://drive.google.com/file/d/1z1Dj0GTUWIp_muC3rA5GGOdOdgaupbDM/view?usp=sharing) | [Volume 33](https://drive.google.com/file/d/1Y_0-Hqjb86l2fB0BJikbsy5eD2JnO_Zi/view?usp=sharing) | [Volume 53](https://drive.google.com/file/d/19fdb4-UrA0_m4cjWft0O64fdViMCBYHQ/view?usp=sharing) | [Volume 73](https://drive.google.com/file/d/1e0mwxaR-S-CStdkvK5u6RpmY74kSc5Bp/view?usp=sharing) |
| [Volume 14](https://drive.google.com/file/d/1hIfAs99c5mb0e6N9Cs321SiIMivw0hkB/view?usp=sharing) | [Volume 34](https://drive.google.com/file/d/1HSmqxOZfhsaHz9_ahwToPKUk0J7GvVI3/view?usp=sharing) | [Volume 54](https://drive.google.com/file/d/1mbWPSXskrH9K3ZhgIM6B01b-QFo6F4gd/view?usp=sharing) | [Volume 74](https://drive.google.com/file/d/1XPVW5wE8SeQsFChNy0LUAbh-M0C72AHl/view?usp=sharing) |
| [Volume 15](https://drive.google.com/file/d/1ajwvfZPGR680ylWhKjWWOIiMIyV7DvdW/view?usp=sharing) | [Volume 35](https://drive.google.com/file/d/1Uvzxzo3E1q2zYTnP3OpoNyw8f09FvXww/view?usp=sharing) | [Volume 55](https://drive.google.com/file/d/1VS4ysEc-xTplFPBp7KTqGnjxlYaJQaU1/view?usp=sharing) | [Volume 75](https://drive.google.com/file/d/1nyVLRQcK84IkOkYfTbz0QLS_Vnjd7l8Q/view?usp=sharing) |
| [Volume 16](https://drive.google.com/file/d/1EwdPExonC1lxYncO3adANwYN56B_YcD0/view?usp=sharing) | [Volume 36](https://drive.google.com/file/d/1Rgbi5T8hsPzl0cBGrlUyymFOxoVaFgaY/view?usp=sharing) | [Volume 56](https://drive.google.com/file/d/1ixpfqk5xrTlaP9f-GhgHstJDnorKyaju/view?usp=sharing) | [Volume 76](https://drive.google.com/file/d/1YHWGULNuEfp8FR2FQWtI7_0wH0HgMvBv/view?usp=sharing) |
| [Volume 17](https://drive.google.com/file/d/1ocjx8b9xfrSObnNO3CyHY6KwOGdllTZF/view?usp=sharing) | [Volume 37](https://drive.google.com/file/d/1ZO0Ns83z3CKBO8Ywv0OK0ORjs0WDIzi_/view?usp=sharing) | [Volume 57](https://drive.google.com/file/d/1yiL035TqzazdXikaNWLQBPl1eIUKCUGz/view?usp=sharing) | Volume 77: May 2021 |
| [Volume 18](https://drive.google.com/file/d/1otDFaUwV_-EwKpHjm3gjgcGeo0zc1C90/view?usp=sharing) | [Volume 38](https://drive.google.com/file/d/1EAfl2B8wiHvCzXoM2NgqNfcReJ_PjF0u/view?usp=sharing) | [Volume 58](https://drive.google.com/file/d/1g1ZCHl7VW6LfqjF7ohjzgo9h5IBW_c-R/view?usp=sharing) |  |
| [Volume 19](https://drive.google.com/file/d/1Wkmt4lZhWXjae500BjFeCetq3Ti9getF/view?usp=sharing) | [Volume 39](https://drive.google.com/file/d/1QjR3ikAj6KbnztoqJlp2l5hxS0coe_y8/view?usp=sharing) | [Volume 59](https://drive.google.com/file/d/1CbhztFSdVgXWfaq50oEy_FRgJTPoOkJw/view?usp=sharing) |  |
| [Volume 20](https://drive.google.com/file/d/1p49--Wp-oR-VRSwhfHB1EvwjapU9YQ5v/view?usp=sharing) | [Volume 40](https://drive.google.com/file/d/1_qX0lp88X8IKedYsn2n-1S5oVKuViDbQ/view?usp=sharing) | [Volume 60](https://drive.google.com/file/d/17gHCipJm0ie3wWwmQVzldDCHvpZGbBY6/view?usp=sharing) |  |