Introduction to the Arduino Microcontroller

This popular and versatile microcontroller could be the brains of your next ham radio project.

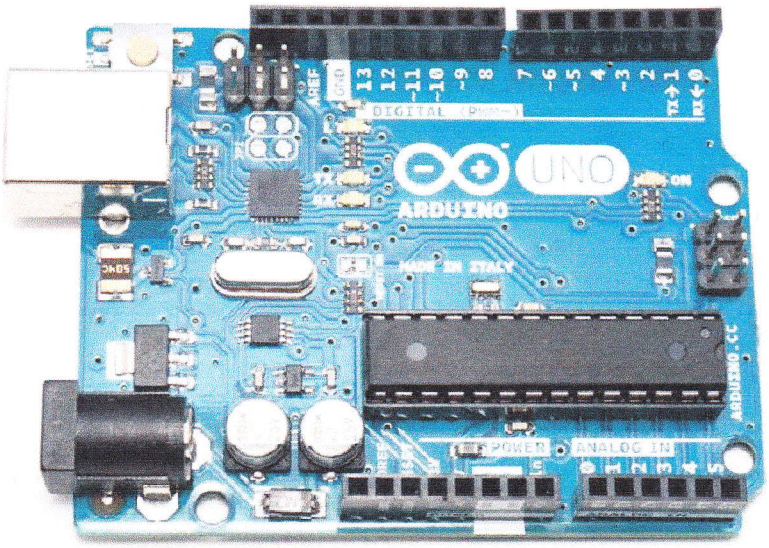
Glen Popiel, KW5GP

Hams get to tinker with all manner of wonderful new technologies. However, quite often it seems the projects we want to build are difficult to construct, too hard to understand, or just too expensive. Often, it's easier to buy a commercial version or a pre-assembled kit. That's the way it was with me before I discovered the Arduino.

Enter the Arduino

The Arduino is a small, inexpensive, easy-to-program microcontroller that is rapidly becoming one of the favorite tools for anyone wanting to build electronic projects, from simple LED and motor/servo controllers, all the way to robotics. Based on the eight-bit 16 MHz Atmel® series of microcontrollers, the Arduino has 14 digital Input/Output (I/O) pins and six 10-bit Analog to Digital (A/D) input pins on a standard board footprint that supports many add-on boards, known as *shields*. The Arduino also supports several industry-standard bus technologies including Serial, Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I^2C), and 1-Wire, which allow you to add hundreds of inexpensive add-on modules.

Released under the *Creative Commons Attribution Share-Alike* licensing model, the Arduino board designs and schematics are completely open source. The open source approach has fostered an entire community of builders and developers freely sharing their ideas and creations, and has led to an explosion of inexpensive Arduino boards and add-on components. The low cost associated with the Arduino allows even the inexperienced builder to create and experiment with near-total abandon, with the cost of recovering from a catastrophic "releasing of the smoke" often less than the pizza you order to console yourself over the loss. I have found the Arduino quite forgiving, and the odds of your project sending smoke signals are slim.



How It All Began

Created in 2005 at the Interaction Design Institute Ivrea in northern Italy, the Arduino was originally intended as an inexpensive microcontroller platform for students, replacing the more expensive and less powerful Parallax "Basic Stamp" microcontroller. From the start, the Arduino was designed for a non-technical audience of artists, designers, and students in the creative fields to develop school projects. An easy-to-use Integrated Development Environment (IDE) was developed, allowing easy programming of the Arduino using a PC, Mac, or Linux workstation. When the Institute closed for lack of funds, the Arduino team released the entire project as open source, making the Arduino one of the first open source hardware projects. The Arduino is easy to learn, and in no time you can be creating fun and interesting projects.

The open source approach has fostered an entire community of builders and developers freely sharing their ideas and creations.

This open source approach has led to the immense popularity of the Arduino. The Arduino and its many variants are now available in many configurations and CPU speeds. The number of shields and modules available for the Arduino is already quite extensive and growing daily. Other manufacturers are creating their own Arduino derivatives, including the Digilent® chipKIT Uno32®, used as the brains for the TEN-TEC Rebel and Patriot open source QRP transceivers.

The Arduino Uno and its newer brother, the Leonardo, are the most popular among builders. A wide variety of 100% Arduino-compatible Uno and Leonardo boards are available from eBay for less than $10. Inexpensive Arduino shields and modules are similarly available from eBay, SparkFun, Adafruit, and many other suppliers. You can buy a complete Arduino development kit for under $100.

General Specifications

The Arduino Uno consists of an eight-bit

16 MHz Atmel ATmega328 microcontroller with 32 KB of flash memory used to hold your Arduino program (known as a *sketch*), 2 KB of static random access memory (SRAM) for variables, and 1 KB of EEPROM (electrically erasable programmable read-only memory) for semi-permanent data storage. The Uno has 14 digital I/O pins and six 10-bit analog to digital (A/D) inputs that can also be used for digital I/O. An on-board voltage regulator allows you use an external 7 to 20 V dc power source. You can also power it from the USB port used to connect the Arduino to a workstation for programming and debugging.

Based on the ATmega32U4, the Leonardo is nearly identical to the Uno, with 2.5 KB of SRAM, improved USB port functionality, and other improvements. Some Arduino variants don't have the onboard power regulator and USB port, but for now we'll stick to the more common Uno-type boards.

Designed for expandability, I/O and power connections are brought out to a series of headers around the edge of the board. The layout of headers is standard among the majority of the Uno-type boards. Shields can be plugged into these headers, and even stacked one on top of the other, providing power and I/O to the shield without any additional wiring.

What It Can Do

As the name "microcontroller" implies, the Arduino is designed to sense and control things. There are all manner of shields and modules available. There are shields with all types of displays, motor and relay drivers, Ethernet, Wi-Fi, packet radio, and much more. There is also a huge selection of Arduino-compatible modules and components. These modules include lightning sensors (see Figure 1), direct digital frequency synthesis (DDS), text-to-speech, voice recognition, GPS, temperature and humidity sensors, and a whole host of other displays, sensors, and modules that can interface to the Arduino.

Programming the Arduino

You can program your Arduino using the free Integrated Development Environment (IDE) that runs on a PC, Mac, or Linux workstation. The IDE supports most of the Arduino variants and includes a text editor for creating and editing your sketches. The IDE is also used to to compile, verify, upload, and debug your sketches via the Arduino onboard USB port. A message area provides feedback for compiler errors and other information. The IDE incorporates a serial monitor for sending diagnostic and debugging information to and from your workstation while your sketch is running. Once uploaded, your sketch runs completely standalone on the Arduino.

Arduino Libraries

Libraries extend the functionality of the IDE, mainly when working with add-

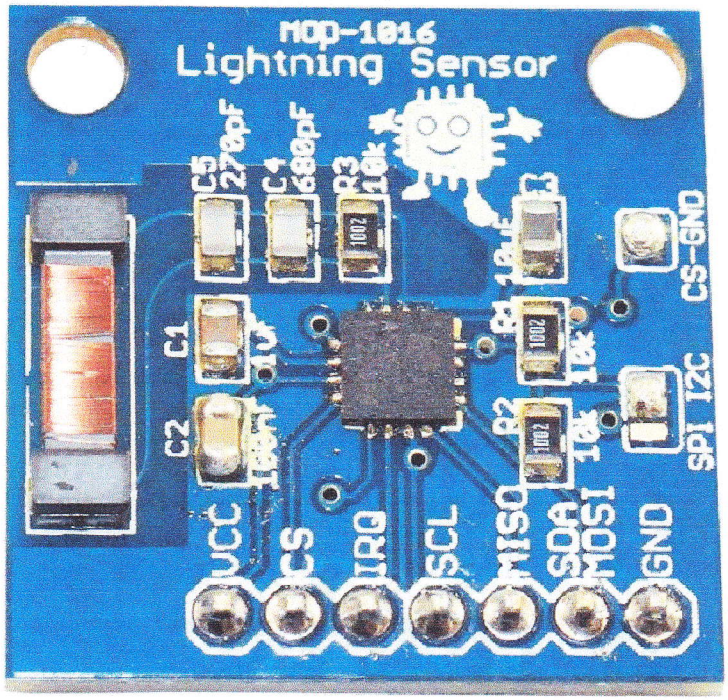


Figure 1 — Lightning sensor module from Embedded Adventures.

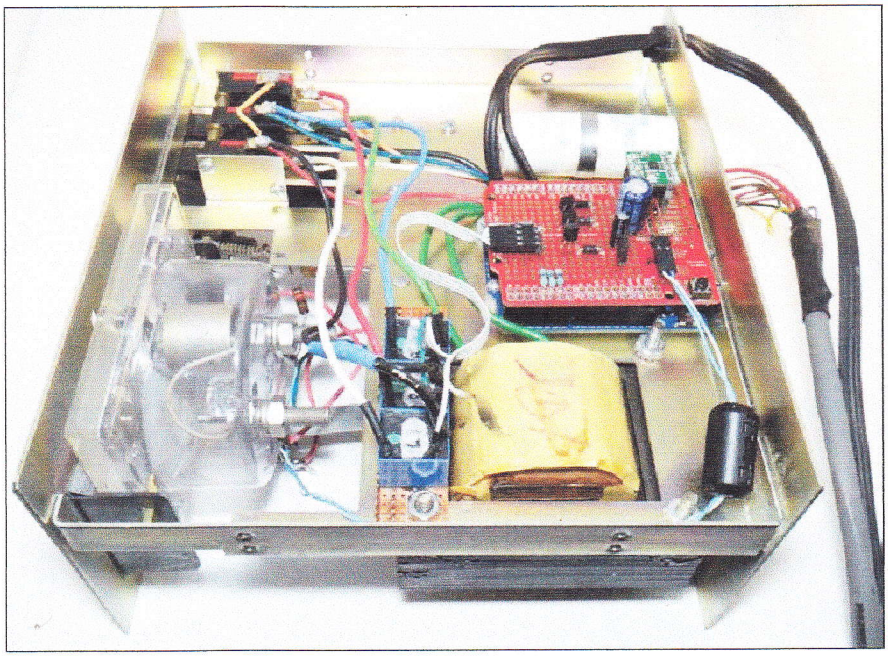


Figure 2 — Modified Ham III rotator controller.

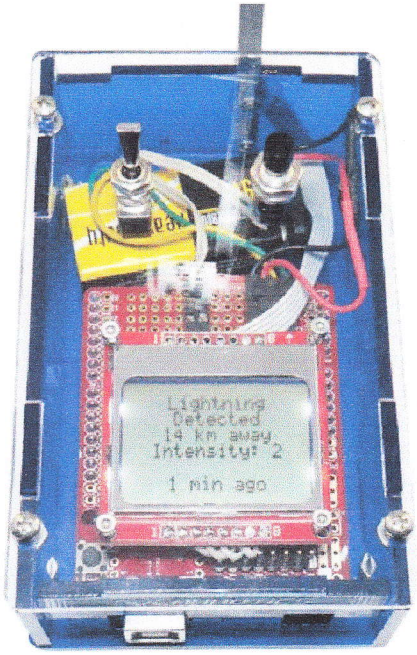


Figure 3 — Arduino-powered lightning detector.

on devices. Think of a library as a set of "software drivers" used to interface with these add-on components. Most Arduino shields and modules have pre-written libraries to support their unique functions. Libraries and their example code can save you hours of research (and frustration) while you are trying to figure out how to get a new shield or module working. The Arduino IDE comes with a number of pre-installed libraries.

Ham Applications

Using the Arduino, you can create a wide variety of ham-related projects. If you enjoy CW, you can use a keyboard and have the Arduino send for you. You can also build a decoder to display received CW on an LCD display. It's possible to use an Arduino to automate your shack. You can modify your manual antenna rotator controller (see Figure 2), and control it using *Ham Radio Deluxe* or other software, and you can do it for under $50.

How about a lightning detector that detects lightning up to 40 km away and have the Arduino automatically disconnect your antennas until the storm passes? You can build one in a matter of hours and it takes just eight wires. An Arduino-powered lightning detector (see Figure 3) helped keep our equipment (and us) safe during Field Day 2014 while we were besieged by strong thunderstorms for nearly the entire weekend.

Using an easily interfaced direct digital frequency synthesis (DDS) module, you can build a precision digitally controlled frequency generator. For only $9, you can get a DDS module that will operate up to 40 MHz. You can build a digital VFO and display for that old "rockbound" rig, or go so far as to build your own transceiver from scratch.

Let's not forget our visually, hearing, and otherwise impaired hams. A text-to-speech module and a voice recognition module (see Figure 4) coupled with the Arduino ability to control things, makes it an ideal platform to help automate and simplify our ham shacks. You can control your transceiver, antenna rotator, and the rest of your shack with the sound of your voice.

In Summary

The Arduino is an inexpensive and powerful tool for the homebrewer. You can now design and build a wide variety of sensing and control applications in a weekend, instead of months. You don't have to be a programming guru to create fun and useful projects for your ham shack. There is an entire community of Arduino developers at your back and some excellent tutorials at **www.arduino.cc**, **www.sparkfun.com**, and **www.adafruit.com**.[1] When it comes to the Arduino, you are limited only by your imagination. Hopefully, you'll find working with the Arduino to be as enjoyable, challenging, and satisfying as I have.

[1]See also, *Arduino for Ham Radio*, ARRL order no. 0161, available from your ARRL dealer, or from the ARRL Store, Telephone toll-free in the US 888-277-5289, or 860-594-0355, fax 860-594-0303; **www.arrl.org/shop/**; **pubsales@arrl.org**.

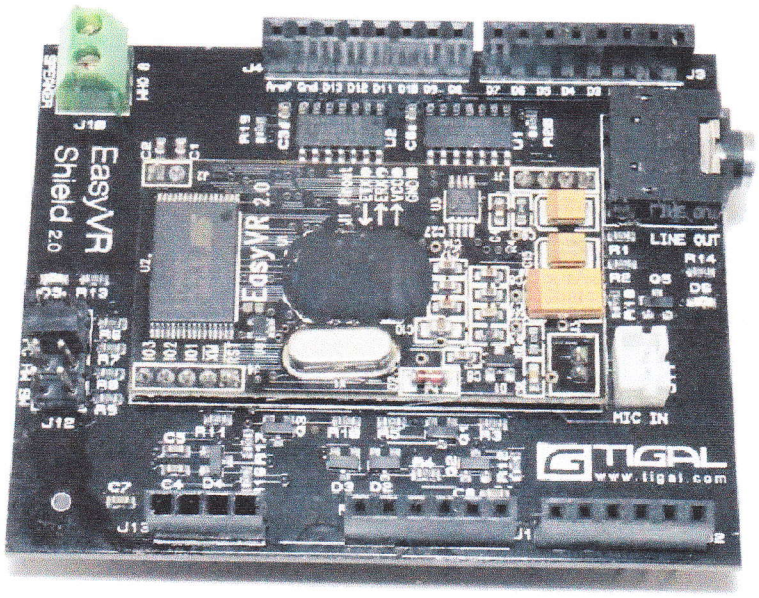


Figure 4 — Easy VR Voice Recognition Shield from VeeaR (**veear.eu**).

All photos by the author.

Glen Popiel, KW5GP, authored the ARRL book, *Arduino for Ham Radio*. He is an ARRL member, Volunteer Examiner, member of QRP-ARCI, and the Olive Branch Amateur Radio Club. By day, he is a Network Engineer and Technology Consultant for Ciber, Inc., and the Mississippi Department of Education, specializing in open source solutions. First licensed in 1972 as WN4FTX, and later WA4FTX, Glen holds an Amateur Extra class license and has worked in the computer and electronics field for over 40 years. Glen is also a member of the QRP SkunkWerks, a design team of fellow hams and Arduino enthusiasts who have succeeded in getting the JT65 digital mode to run natively on the TEN-TEC Rebel CW-only QRP transceiver. He currently lives in Southaven, MS, where he continues to develop new Arduino projects. You can reach Glen at **kw5gp@arrl.net**.

For updates to this article, see the *QST* Feedback page at www.arrl.org/feedback.