How to Use an Arduino

By Vivian Law

Introduction

The first microcontroller, TMS-1802-NC, was built in 1971 by Texas Instruments. It owed its existence to the innovation and versatility of silicon and the development of integrated circuits (ICs). The microcontroller is a tiny, but powerful computer that can be programmed to perform many tasks that a standard computer would do, but with a fraction of the size and power.

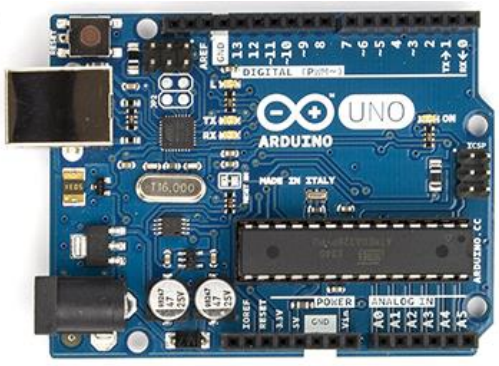
The key components of a microcontroller include:

1. A central processing unit (CPU): The main core of a computer where every command runs through and is routed to its appropriate location for execution.
2. Memory:
    a. Random access memory (RAM): It is 'random' because the data is accessed in a random manner, instead of an arranged or set order. There is a fixed amount of RAM in a microcontroller.
    b. Read-only memory (ROM): This also permits random data access, but the data stored here cannot be changed or modified. It can only be 'read' or displayed.
3. Input/ Output (I/O): The interface between the computer and the real world.

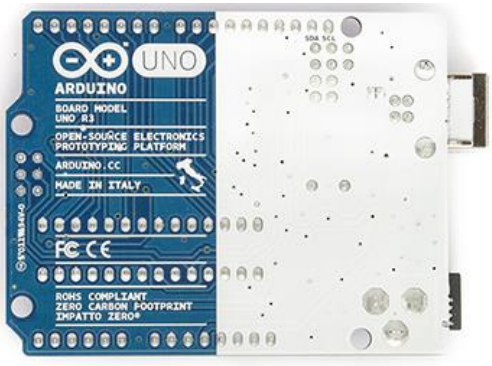
These and other components are all embedded onto a single board to make a microcontroller.

Currently, there are a number of microcontrollers on the market, including but not limited to: Raspberry Pi, TI MSP430 Launchpad, Atmel AVR, and the Arduino Uno. The Arduino Uno will be picked for this tutorial because of its affordability, simple development environment, and the vast amount of resources available online.

What is the Arduino Uno?



Arduino Uno R3 Front



Arduino Uno R3 Back

The Arduino Uno is a microcontroller based on the ATmega328 chip. It has 14 digital I/O pins (six can be used as pulse width modulation (PWM) outputs), six analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button.

Power

Power can be provided either through a USB connection or through an external power source, like a wall-wart or a battery. The recommended range is 7 to 12 volts.

Several power pins exist on the microcontroller board. They are outlined below.

- Vin: Access to the input voltage of the board. Power can be supplied through this port, or accessed through here if powered via the power jack.
- 5V: Outputs a regulated 5 volts from the regulator on the board.
- 3V3: Outputs a regulated 3.3 volts from the regulator on the board.
- GND: Ground pins.
- IOREF: Provides the voltage reference with which the microcontroller operates.

I/O

Each of the 14 digital pins can be used as either input or output using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. Pins with specialized functions are detailed below.

- Serial - 0 (RX) and 1 (TX): Used to receive (RX) and transmit (TX) transistor-transistor logic (TTL) serial data. Serial communication at a TTL level uses a logic high (‘1’) represented by Vcc and a logic low (‘0’) represented by 0V. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts – 2 and 3: Can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. Check attachInterrupt() function for more details.
- PWM – 3, 5, 6, 9, 10, and 11: Provide 8-bit PWM output with analogWrite() function. Pulse width modulation varies how much time the signal is high and low. In other words, the proportion of time the signal is high compared to low over a time interval can be changed using PWM.
- SPI – 10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK): Support serial peripheral interface (SPI) communication. These are used to send data between the microcontroller and small peripherals such as shift registers, sensors, and SD cards.
- LED – 13: Built-in LED connected to pin 13. When the pin is HIGH, the LED is on. When the pin is LOW, it is off.

There are six analog inputs, A0 through A5, each providing 10 bits of resolution (1024 different values). They measure from ground to 5 volts. Pins with specialized functions are detailed below.

- TWI (Two wire interface) – A4/ SDA pin or A5/ SCL pin: TWI is similar to I2C and the bus is identical to that of I2C. Support TWI communications using the Wire library.
- AREF: Reference voltage for the analog inputs. The analogReference() function is used with this pin.

- Reset: Bring this low to reset the microcontroller. Typically used to add a reset button to shields that block the original button on the board.

Programming

The Arduino Uno is programmed through the Arduino software. Select "Arduino Uno" from Tools > Board menu. See Figure 1.

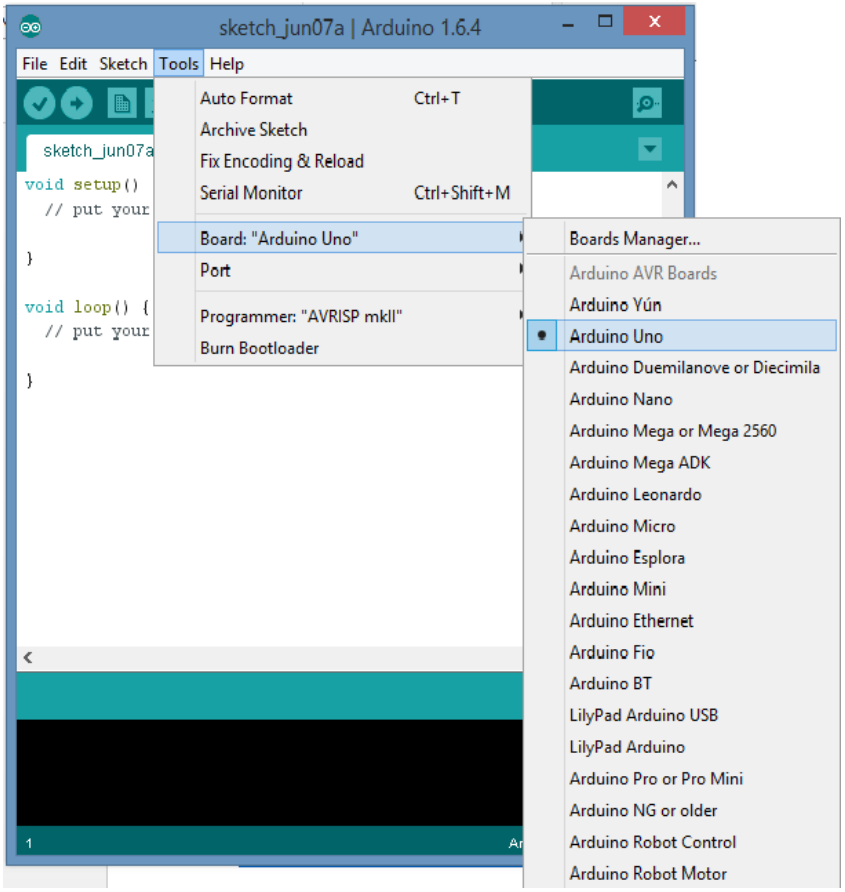


Figure 1: Select Arduino Uno from menu option

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code without the use of an external hardware programmer. A bootloader is a program that loads an operating system when the microcontroller turns on.

The Arduino environment uses its own Arduino language that is based off of C/C++ language.

Soft Reset

The Arduino Uno utilizes 'soft resets,' which are resets of the hardware using software instead of a physical switch or button. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 through a 100 nF capacitor. When this line is low, the reset line drops long enough to reset the chip. The Arduino software utilizes this to upload code when the user presses 'upload' in the Arduino environment. This allows the bootloader to have a shorter timeout.

For advanced users, the Arduino Uno contains a trace on the board that can be cut to disable the auto-reset function that occurs every time the board is connected to the computer. The pads on either side of the trace can be soldered together to re-enable it. It is labeled "RESET-EN." You can also disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line. See Figure 2.

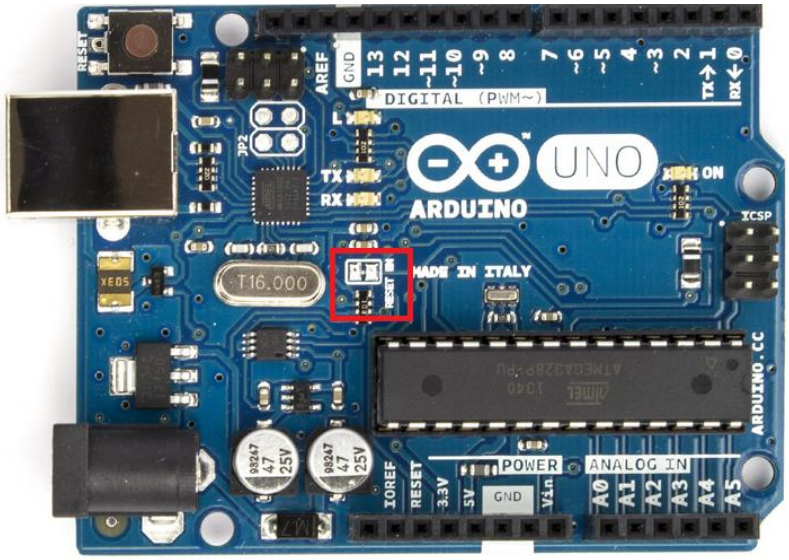


Figure 2: RESET-EN is located within the red box

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects the computer's USB ports from shorts and overcurrent. Even though most computers have internal protection, this fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will break and the connection will be shorted until the overload is removed.

First Sketch

A sketch it what a program is called in the Arduino environment. This is what will be uploaded and run on any Arduino board.

The first sketch that is usually run to demonstrate ease of use for the board is to make an LED blink on the board.

The following code can be found on the Arduino website under Tutorial > Blink.

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Comments are denoted with /*, */, or //. These make the code easier to read and understand, which is especially helpful if the code is shared and utilized between many parties.

The first actual line of the code is “int led = 13.” This line is initializing pin 13 and giving it a name to reference to in the rest of the code.

The section “void setup()” is always included in a brand new sketch. This is only run once in order to set up the rest of the sketch. pinMode() is a built-in function and it is used to initialize pin 13 (AKA led) as an output.

The section “void loop()” is also included in a brand new sketch. This allows whatever code is inside to run infinitely unless there is an interrupt or break condition set. As mentioned in the code comments, the LED is turned on with a HIGH value, asked to hold this value for one second, and then is turned off using a LOW value. This is also held for one second, before repeating infinitely.

Done correctly, the LED will blink continuously until power is disconnected. See Figure 3.

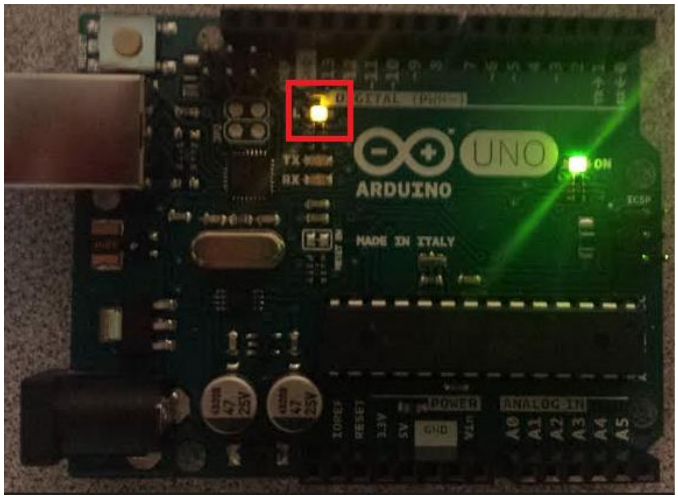


Figure 3: Blinking LED is encased in the red square. The green LED is for power.

Additional Basic Sketches

Analog Read Serial

This sketch demonstrates how to read analog input using a potentiometer. A potentiometer, or a pot, provides a varying amount of resistance when the knob is turned. To measure the resistance of the pot, we can pass through voltage through and utilize Ohm's Law, V=IR, to calculate the resistance, in which V equals voltage, I equals current, and R equals resistance. We will then monitor the state of the potentiometer by establishing serial communication between the Arduino Uno and a computer.

Besides the Arduino Uno, we will also need some external hardware: 10 kOhm potentiometer.

Potentiometers come in all different shapes, as we can see in Figure 4 below. It is not important what type of potentiometer we use in this experiment, but for clarity, the tutorial will show an example of the precision shaft potentiometer.

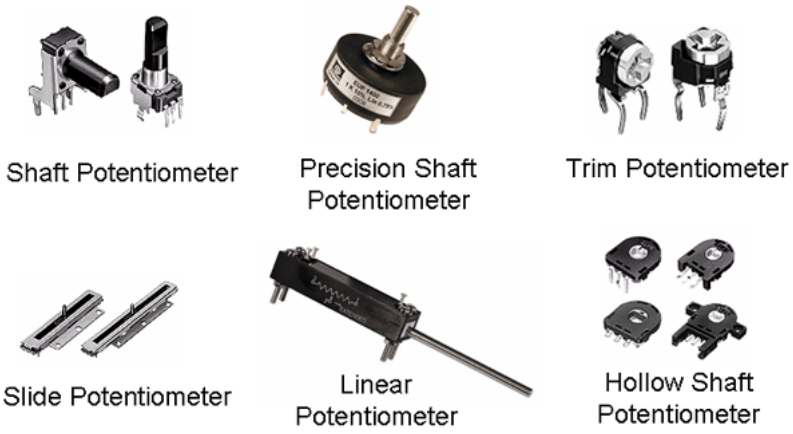


Figure 4: Different types of potentiometers

The potentiometer has three wires, two for the ends of the potentiometer and one for the movable leg. Measuring from the two ends, there is a fixed amount of resistance. In this case, it will be 10 kOhm. The third leg determines the varying resistance. Measuring from one of the fixed legs to the varying leg will give a varying resistance as the knob is turned.

Now that we understand potentiometers, we can start the experiment! Connect the three wires of the potentiometer to the Arduino Uno. Pick one of the outer pins to put to ground. The second goes from 5 volts to the other outer pin. The third goes from analog input 0 to the middle pin of the potentiometer. See Figure 5.

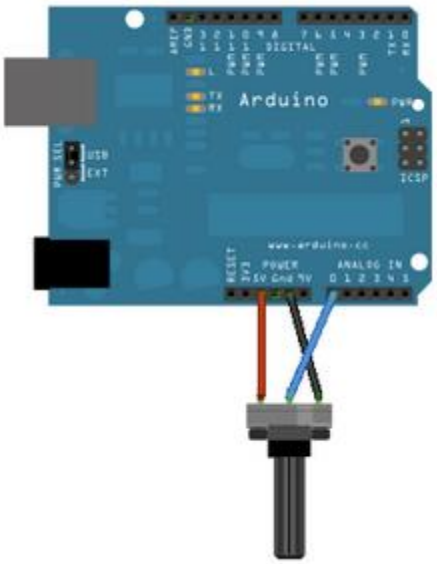


Figure 5: Picture of the potentiometer connected to the Arduino Uno

As we turn the knob, the resistance from the potentiometer changes and the analog voltage we are using as input changes as well. The Arduino Uno has an analog-to-digital converter that reads the changing voltage and converts it to a number from 0 to 1023. When the knob is turned all the way to one side, there will be 0 volts going to the pin. So the value will be 0. When the knob is turned all the way to the opposite side, there will be 5 volts going to that pin. So the value will be 1023. The analogRead() function returns a number between 0 to 1023 that is proportional to the amount of voltage at the pin.

```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

 This example code is in the public domain.
 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

For the setup, we only need to begin serial communications between the Arduino Uno and the computer. For this, we will be using the 9600 baud rate. Baud rate is the bits of data per second that is being transmitted between the microcontroller and the computer. This is done with the Serial.begin(9600) function call.

In the main loop of the sketch, we need to establish a variable to store the values of resistance from the potentiometer. We simply initialize an 'integer' datatype with "int sensorValue = analogRead(A0)." Then we need to print this information as a decimal to the output screen using Serial.printIn(sensorValue, Dec).

The sketch is now complete and can be uploaded to the Arduino Uno.

Now, open the Serial Monitor in the Arduino environment to view the data streaming from the potentiometer. See Figure 6 and 7.

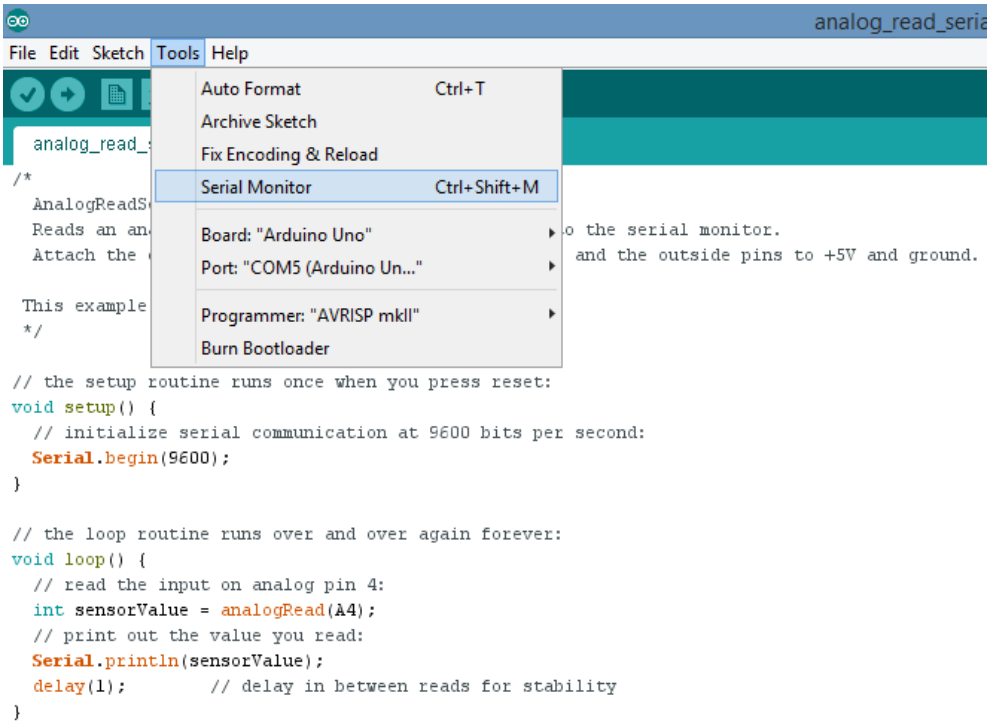


Figure 6: Serial Monitor option shown in the menu selection

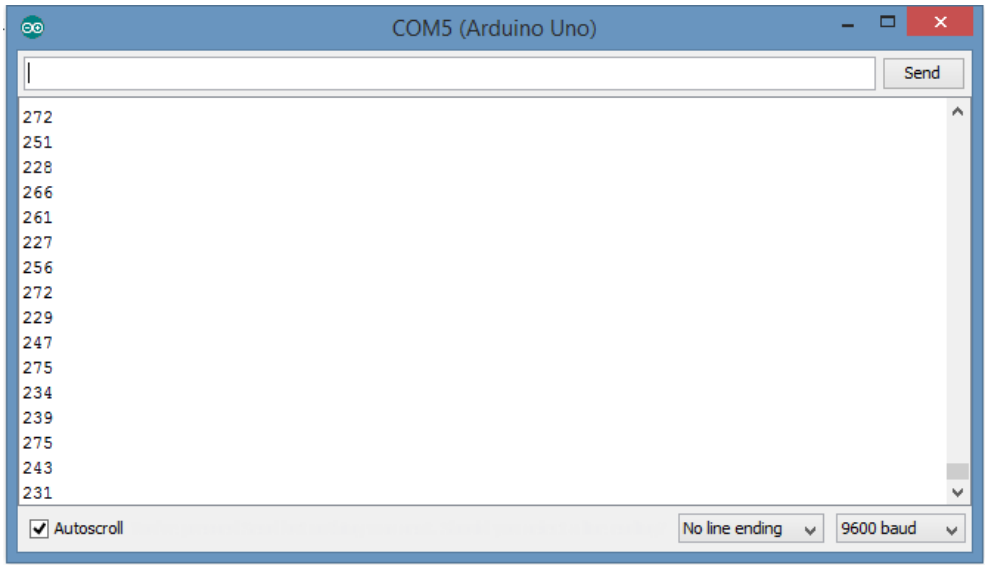


Figure 7: Serial Monitor with live data streaming

The values directly correlate to the resistance of the potentiometer, so as we turn the knob, the numbers will change almost instantaneously.

Summary

Microcontrollers have allowed hobbyists and first time programmers to easily pick up the fun and exciting activity of hardware programming. Electronic projects that used to be intimidating and take hours upon hours of coding now take only 30 minutes or less to put together. The Arduino Uno and many other microcontrollers of its generation have an extended community dedicated to the sharing of fun projects and the blocks of code underlying them. Among microcontrollers, the Arduino Uno is known to be one of the easiest for beginners with little or no experience to just pick up and begin to build fun projects. Hopefully this short tutorial has helped and inspired more people to venture into the fun realm of electronics.

References

"Arduino Uno." *Arduino*. N.p., 2015. Web. 8 June 2015.

Choudhary, Himanshu. "Difference Between Microprocessor and Microcontroller." *Engineers Garage*. N.p., n.d. Web. 8 June 2015.

Grusin, Mike. "Serial Peripheral Interface (SPI)." *Sparkfun*. N.p., n.d. Web. 8 June 2015.

Jimb0. "RS-232 vs. TTL Serial Communication." *Sparkfun*. N.p., 23 Nov 2013. Web. 8 June 2015.

Jordandee. "Pulse-width Modulation." Sparkfun. N.p., n.d. Web. 8 June 2015.

"Microcontroller – Invention History and Story Behind the Scenes." *CT Circuits Today*. N.p., 23 Oct 2013. Web. 8 June 2015.

Westfw. "How to choose a Microcontroller." *Instructables*. N.p., n.d. Web. 8 June 2015.

"What is the difference between I2C and TWI?" *Stack Exchange*. N.p., 10 Dec 2012. Web. 8 June 2015.

Wonder, Dan. "What do RAM & ROM Mean?" *Chron*. Demand Media, n.d. Web. 8 June 2015.