SparkFun Joystick Shield

For the Arduino Uno

Implemented by: Joseph Robinson, Angelo Gonzales, William Black, and Pascal Gautam

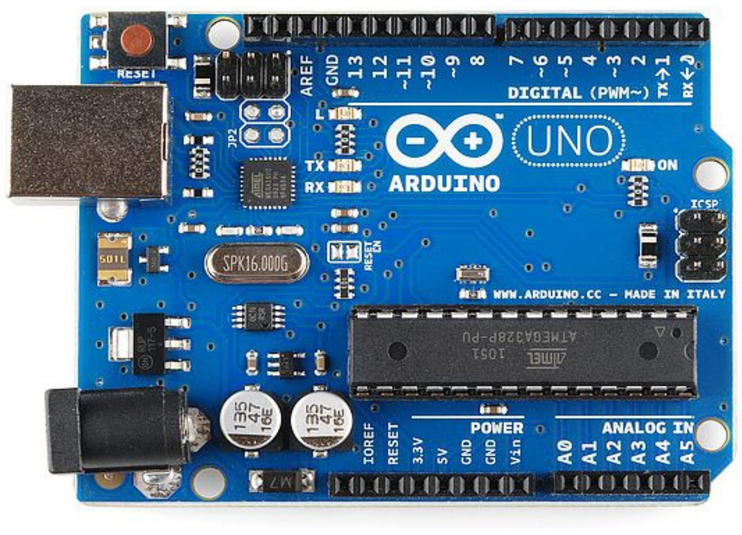
Table of Contents

Introduction

AVR microcontrollers were originally introduced by Atmel (now owned by Microchip Technology) in 1996. This family of microcontrollers was special at the time of release because they used flash memory on the chip to store program data rather than older EEPROM standards.

AVR chips are found in an almost-endless list of different embedded devices at the industry level, but Arduino brought AVR to hobbyists and educators. Arduino provides a low cost-of-entry family of microcontrollers to their customers as well as free tutorials and a powerful IDE.

The Arduino Uno



The controller given to our group was the Arduino Uno. The Arduino Uno is a micro-controller board which is based on a removable dual-inline-package (DIP) ATmega328 AVR microcontroller. It has 20 digital input and output pins of which 6 can be used as PWM outputs and 6 can be used as analog inputs. It has 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset Button. It contains everything that is essential to support the microcontroller. Below are some relevant technical details.

Microcontroller	ATmega328P
Operating Voltage	5V
Analog Input Pins	6
Digital I/O Ports	14 Total; 6 with Pulse Width Modulation
Flash Memory	32 KB on chip with 0.5 KB reserved for the bootloader

Clock Speed	16 Mhz

The Uno’s USB port can both provide enough power to run the controller and be used to write new program data to the flash memory quickly.

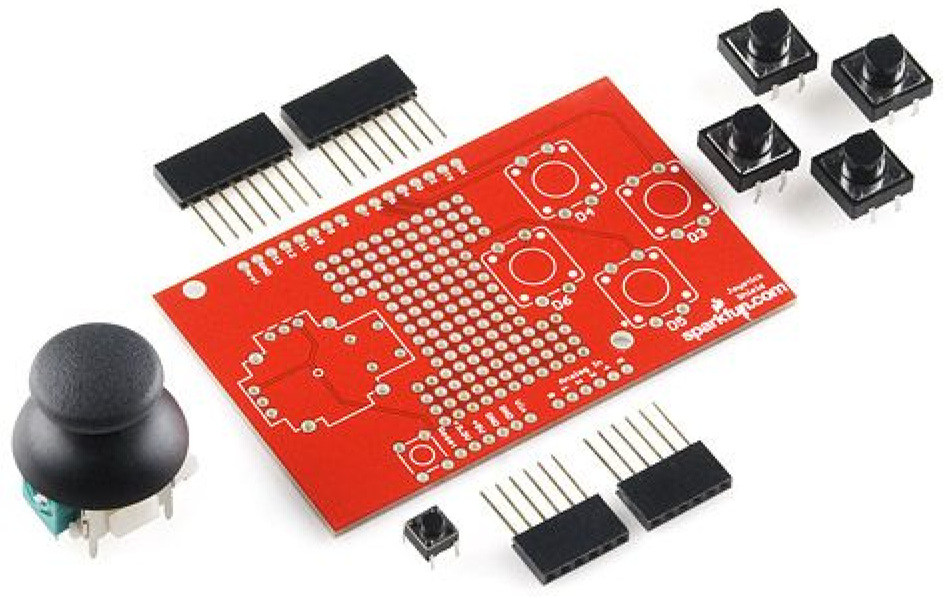
Arduino board is getting popular among students, hobbyists, artists, programmers and other different professionals. It has been the brain of thousands of projects, from day to day hacks to complex scientific inventions. The main reason behind this is that it is easy to use for beginners and flexible for expert users. Also, it runs on all three major operating systems Mac, Linux and Windows.

The Joystick Shield

SparkFun Electronics sells a series of different “shield” devices. Shields are secondary (or tertiary, if you’re brave enough to keep stacking) printed circuit boards (PCB) that can be easily slotted into the top of your Uno board.

The Joystick Shield comes with an analog thumbstick and six buttons. The thumbstick uses two analog ports (0 and 1) to report its position data. The stick has two potentiometers, oriented perpendicularly, each reporting a value between 0 and 1023. These two values can be used as X and Y positions on a coordinate plane to visualize the range of motion.

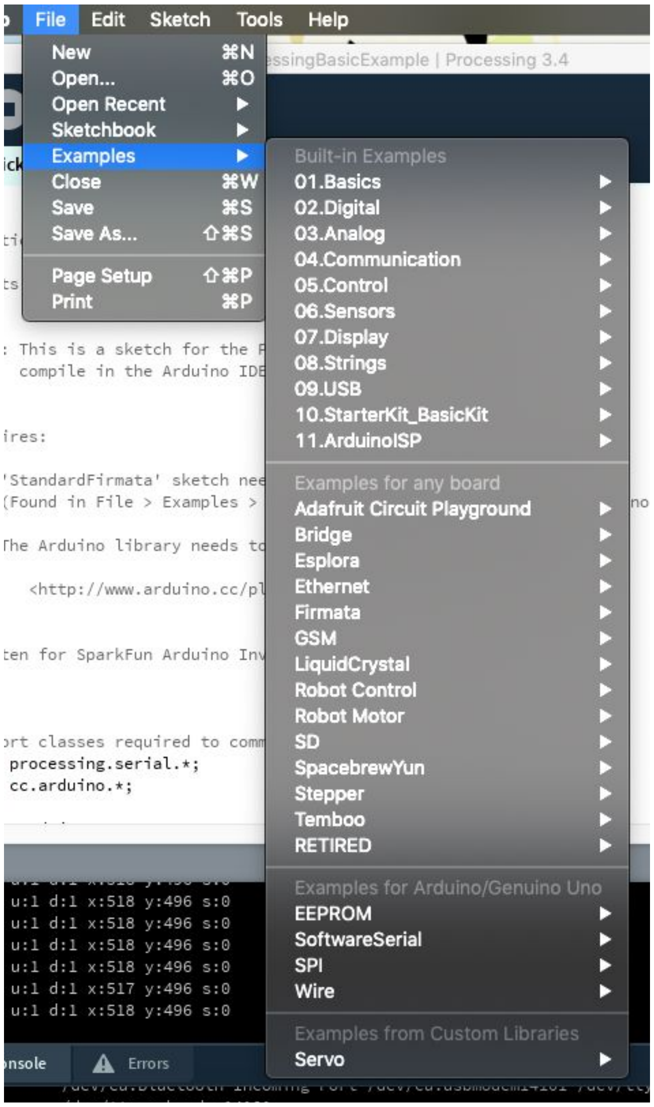
The four large, obvious buttons each report to a digital pin. They range over from ports 3 to 6. The thumbstick can be pushed down to click a fifth input button connected to digital port 2. The final button included in the kit is a reset button. This is a smart idea as larger shields might cover the entirety of the Uno.



Development Tools

The Arduino IDE

The most simple option for starting AVR development is the Arduino IDE. This program allows coding in C through the use of numerous libraries. These libraries range from simple to complex and the IDE comes with dozens of example files to help get you started.



```
: This is a sketch for the P
  compile in the Arduino IDE

ires:

'StandardFirmata' sketch nee
(Found in File > Examples >

The Arduino library needs to

   <http://www.arduino.cc/pl

ten for SparkFun Arduino Inv

ort classes required to comm
 processing.serial.*;
 cc.arduino.*;
```

```
u:1 d:1 x:518 y:496 s:0
u:1 d:1 x:518 y:496 s:0
u:1 d:1 x:518 y:496 s:0
u:1 d:1 x:518 y:496 s:0
u:1 d:1 x:517 y:496 s:0
u:1 d:1 x:518 y:496 s:0
```

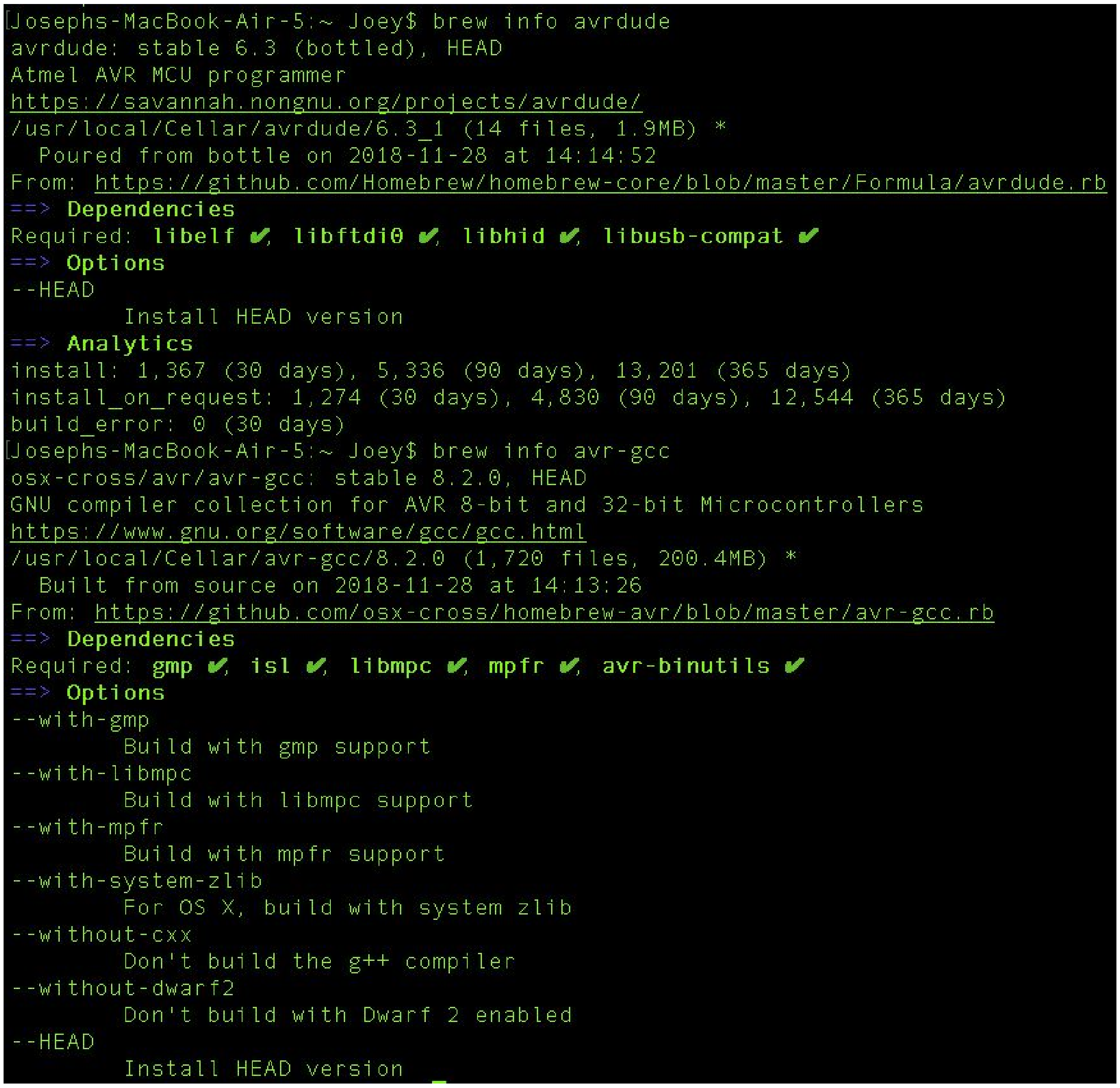
The Arduino IDE was written in Java and based on Processing. Processing is a self-described “software sketchbook” with the goal of creating interdisciplinary works between visual artists and programmers.

Command Line Tools

AVRDUDE and avr-gcc are your main tools when working outside the standard Arduino IDE. AVRDUDE is maintained by www.nongnu.org users and the history is described on their site as follows:

> “AVRDUDE has once been started by Brian S. Dean as a private project of an in-system programmer for the Atmel AVR microcontroller series, as part of the Opensource and free software tools collection available for these controllers. Originally, the software was written for the FreeBSD operating system, maintained in a private CVS repository, and distributed under the name avrprog.
>
> Due to the growing interest in porting the software to other operating systems, Brian decided to make the project publically accessible on savannah.nongnu.org. The name change to AVRDUDE has been chosen to resolve the ambiguity with the avrprog utility as distributed by Atmel together with their AVRstudio software.”
>
> Retrieved from https://www.nongnu.org/avrdude/ on 12 Dec 2018.

avr-gcc is the GNU’s compiler collection for 8- and 32-bit microcontrollers. Both command line tools can be easily installed on macOS using Homebrew (the OSX answer to apt).

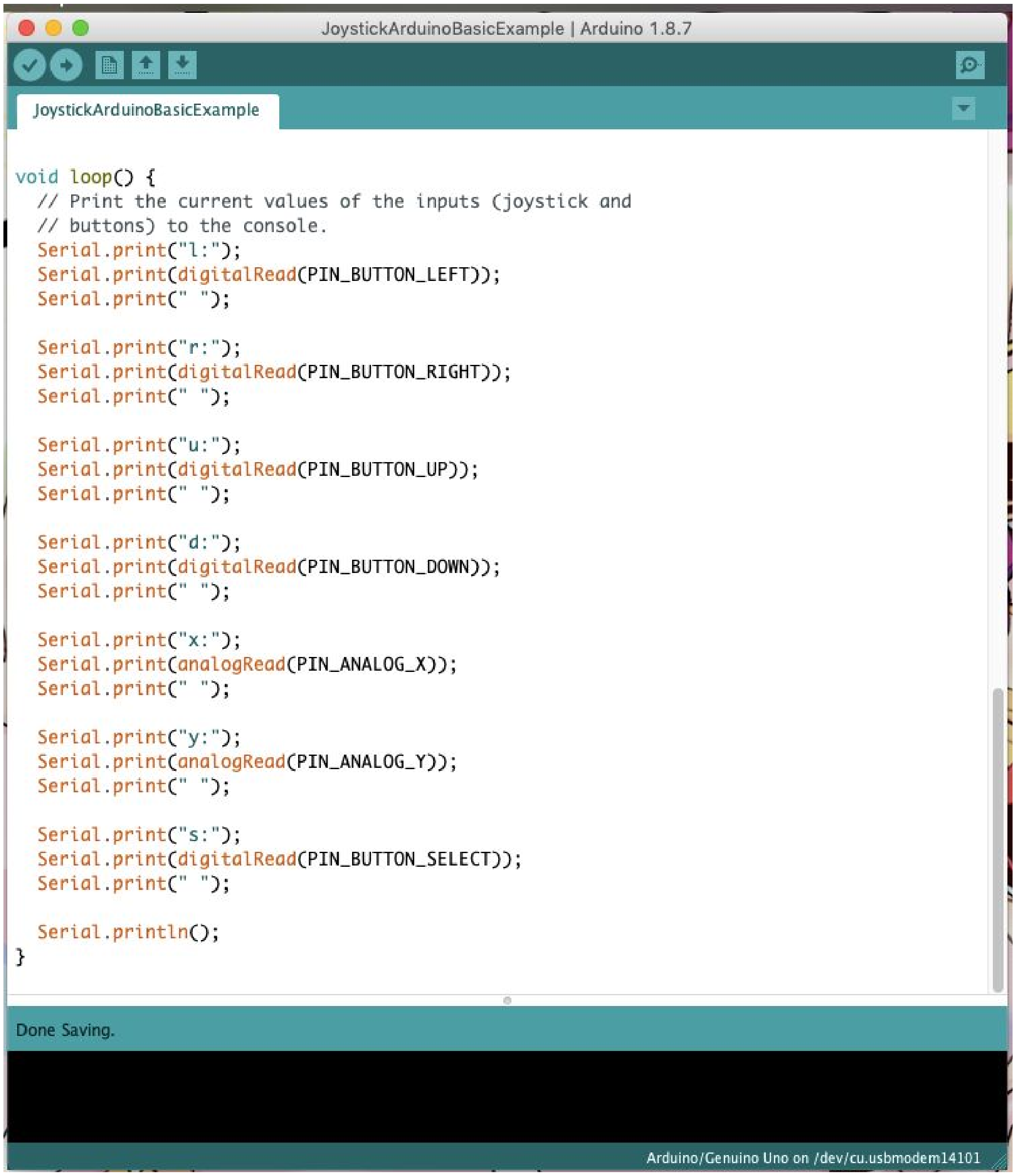


```
Josephs-MacBook-Air-5:~ Joey$ brew info avrdude
avrdude: stable 6.3 (bottled), HEAD
Atmel AVR MCU programmer
https://savannah.nongnu.org/projects/avrdude/
/usr/local/Cellar/avrdude/6.3_1 (14 files, 1.9MB) *
  Poured from bottle on 2018-11-28 at 14:14:52
From: https://github.com/Homebrew/homebrew-core/blob/master/Formula/avrdude.rb
==> Dependencies
Required: libelf ✔, libftdi0 ✔, libhid ✔, libusb-compat ✔
==> Options
--HEAD
        Install HEAD version
==> Analytics
install: 1,367 (30 days), 5,336 (90 days), 13,201 (365 days)
install_on_request: 1,274 (30 days), 4,830 (90 days), 12,544 (365 days)
build_error: 0 (30 days)
Josephs-MacBook-Air-5:~ Joey$ brew info avr-gcc
osx-cross/avr/avr-gcc: stable 8.2.0, HEAD
GNU compiler collection for AVR 8-bit and 32-bit Microcontrollers
https://www.gnu.org/software/gcc/gcc.html
/usr/local/Cellar/avr-gcc/8.2.0 (1,720 files, 200.4MB) *
  Built from source on 2018-11-28 at 14:13:26
From: https://github.com/osx-cross/homebrew-avr/blob/master/avr-gcc.rb
==> Dependencies
Required: gmp ✔, isl ✔, libmpc ✔, mpfr ✔, avr-binutils ✔
==> Options
--with-gmp
        Build with gmp support
--with-libmpc
        Build with libmpc support
--with-mpfr
        Build with mpfr support
--with-system-zlib
        For OS X, build with system zlib
--without-cxx
        Don't build the g++ compiler
--without-dwarf2
        Don't build with Dwarf 2 enabled
--HEAD
        Install HEAD version
```

Here is the “brew info” data for both programs.

Experiments

The first experiments with the board were simple. SparkFun's product page has links to tutorials for quickly setting up a program to report thumbstick position and button presses using the Arduino IDE.


JoystickArduinoBasicExample | Arduino 1.8.7
JoystickArduinoBasicExample

```
void loop() {
  // Print the current values of the inputs (joystick and
  // buttons) to the console.
  Serial.print("l:");
  Serial.print(digitalRead(PIN_BUTTON_LEFT));
  Serial.print(" ");

  Serial.print("r:");
  Serial.print(digitalRead(PIN_BUTTON_RIGHT));
  Serial.print(" ");

  Serial.print("u:");
  Serial.print(digitalRead(PIN_BUTTON_UP));
  Serial.print(" ");

  Serial.print("d:");
  Serial.print(digitalRead(PIN_BUTTON_DOWN));
  Serial.print(" ");

  Serial.print("x:");
  Serial.print(analogRead(PIN_ANALOG_X));
  Serial.print(" ");

  Serial.print("y:");
  Serial.print(analogRead(PIN_ANALOG_Y));
  Serial.print(" ");

  Serial.print("s:");
  Serial.print(digitalRead(PIN_BUTTON_SELECT));
  Serial.print(" ");

  Serial.println();
}
```

```
Done Saving.
```

```
Arduino/Genuino Uno on /dev/cu.usbmodem14101
```

This first experiment sets up digital pins 2 through 6 and analog pins 0 and 1 to receive input. The main loop just prints a line to the console reporting the input data.

The second basic example uses the Arduino IDE once again to modulate a tune. The tone is output to a speaker via digital pin 9. For the thumbstick, analog pins 0 and 1 are again used. The main loop polls the X and Y position of the stick and uses that information to change the tune. The X position controls the relative tempo of the melody, and the Y position changes the pitch.

The next two projects on the SparkFun page use the StandardFirmata example code. Firmata gives Arduino users a standard library for serial communication. The experiment itself runs in Processing.