Woolsey Workshop

Cheatsheet: Arduino Basics

General

Resources

- Arduino website
- Arduino Products (contains Overview, Tech Specs, and general documentation)
- Arduino Getting Started
- Arduino IDE
- Arduino Sketch Language Reference
- Arduino Tutorials
- Arduino Forum
- Woolsey Workshop Arduino Tutorials

General Information

- Default location for project files
  - Linux/macOS: `~/Documents/Arduino`
  - Windows: `C:\Users\<username>\Documents\Arduino`
- The last program run on the Arduino board is saved in on-board memory and will run automatically when power is applied to the board.

Arduino IDE Navigation

- Select development board
  - Main Menu > Tools > Board:"…" > …
- Select serial port
  - Main Menu > Tools > Port > …
- Create a new program
  - Main Menu > File > New
- Open an existing program
  - Main Menu > File > Open…
- Open an example program (contains basic and library specific examples)
  - Main Menu > File > Examples > …
- Save a new program
  - Main Menu > File > Save As…
- Save an existing program
  - Main Menu > File > Save
  - Save button in window tool bar
- Close a program
  - Main Menu > File > Close
- Print a program
  - Main Menu > File > Print
- Find a word or phrase
  - Main Menu > Edit > Find…

- Include a library
  - Main Menu > Sketch > Include Library > …
- Manage libraries
  - Main Menu > Tools > Manage Libraries…
  - Main Menu > Sketch > Include Library > Manage Libraries…
- Open the Serial Monitor
  - Main Menu > Tools > Serial Monitor
- Get help documentation
  - Main Menu > Help > …

Operation

- Compile and verify sketch
  - Main Menu > Sketch > Verify/Compile
  - Verify button in window tool bar
- Upload and run sketch on board
  - Main Menu > Sketch > Upload
  - Upload button in window tool bar

Start-Up

1. Complete all circuit wiring.
2. Connect the Arduino board to the computer and/or power.
3. Open the Arduino IDE application.

Shutdown

1. Save the current sketch.
2. Run the *BareMinimum* sketch, located at Main Menu > File > Examples > 01.Basics > BareMinimum, to reset all pins back to their default state. Although this is not technically necessary, it is still a good habit to follow in order to avoid accidental damage to the Arduino board.
3. Exit the Arduino IDE application.
4. Disconnect the Arduino board from the computer and power.

Programming

- The programming language used by the Arduino IDE is Sketch that is based on C++ and allows for the use of standard C and C++ library routines.
- A file containing a Sketch program is called a sketch and has a .ino file extension.
- Sketches are saved as a project directory containing a source file with the same name as the project and a .ino file extension. For example, a project named *MyProgram* will be saved in a directory named *MyProgram* that contains the sketch source file named *MyProgram.ino*.
- The `setup()` function runs only once and is where all initialization code in placed.
- The `loop()` function runs repeatedly after the `setup()` function has completed and is where code that is always in action is placed.

- Printing with `Serial.print()` and `Serial.println()` will be shown in the *Serial Monitor* which is displayed from Main Menu > Tools > Serial Monitor.
- Utilize a preprocessor directive to distinguish between different boards, for example, when attaching an interrupt service routine.

```
#ifdef ARDUINO_AVR_UNO_WIFI_REV2
   attachInterrupt(button, resetCount, FALLING);
#else
   attachInterrupt(digitalPinToInterrupt(button), resetCount, FALLING);
#endif
```

- Be careful using a lot of long strings in your program as they can take up a lot of the available memory. If you don't need to modify the strings or data while your sketch is running, you can store them in flash (program) memory instead of SRAM. Do this by using the `PROGMEM` keyword when storing or the `F()` macro when printing.

```
const static PROGMEM char long_str[] = "This is a long string …";
Serial.print(F("This is a long string …"));
```

Example Blink Program

```
// Blink
//
// Description:
// Arduino Sketch program to blink an LED.
//
// Created by John Woolsey on 05/30/2018.
// Copyright © 2018 Woolsey Workshop.  All rights reserved.


const byte redLED = 2;          // reference digital pin 2 as redLED

void setup() {
   pinMode(redLED, OUTPUT);     // set redLED (2) pin as an output

}

void loop() {
   digitalWrite(redLED, HIGH);  // turn on LED
   delay(500);                  // wait 500 milliseconds
   digitalWrite(redLED, LOW);   // turn off LED
   delay(500);                  // wait 500 milliseconds
}
```

Arduino Uno Specific

General

- Digital pins 0 (*RX*) and 1 (*TX*) are connected to the USB on the classic Arduino Uno boards. Do not use when using the USB or are connected to the Arduino IDE as it will interfere with proper USB operations. The Arduino Uno WiFi Rev2 board does not have this limitation.
- The sum of all currents out of all input/output pins combined must not exceed 200 mA. If more current is required on an output, a transistor can be used to drive the component powered by the 5 V pin.
- Pins generally default to inputs in a high impedance state.

Power Pins

- 5 V rated at 500 mA for USB only or 1 A with external supply
- 3.3 V rated at 50 mA

Digital I/O Pins

- Rated 5 V @ 20 mA (absolute maximum of 40 mA)
- Standard TTL compatible levels
- Tri-stated when reset

Analog Input Pins

- Analog reference is 5 V by default, but can be changed via *VREF* pin or by using the `analogReference()` function.
- 10-bit resolution (0 - 1023)
- Tri-stated when reset

Pin-Out

- **3V3** - 3.3 V Power Supply
- **5V** - 5 V Power Supply
- **GND** - Ground
- **GND** - Ground
- **AREF** - Analog Reference Voltage
- **D0/RX** - Digital I/O 0 / SCI Receive (do not use when using USB port with classic Uno)
- **D1/TX** - Digital I/O 1 / SCI Transmit (do not use when using USB port with classic Uno)
- **D2/EI0** - Digital I/O 2 / External Interrupt 0
- **D3/EI1** - Digital I/O 3 (PWM capable) / External Interrupt 1
- **D4** - Digital I/O 4
- **D5** - Digital I/O 5 (PWM capable)
- **D6** - Digital I/O 6 (PWM capable)
- **D7** - Digital I/O 7
- **D8** - Digital I/O 8
- **D9** - Digital I/O 9 (PWM capable)
- **D10/SS** - Digital I/O 10 (PWM capable) / SPI SS
- **D11/MOSI** - Digital I/O 11 (PWM capable) / SPI MOSI
- **D12/MISO** - Digital I/O 12 / SPI MISO
- **D13/SCK/LED_BUILTIN** - Digital I/O 13 / SPI SCK / On Board LED
- **A0** - Analog Input 0
- **A1** - Analog Input 1
- **A2** - Analog Input 2
- **A3** - Analog Input 3
- **A4/SDA** - Analog Input 4 / TWI I2C SDA
- **A5/SCL** - Analog Input 5 / TWI I2C SCL
- **IOREF** - I/O Reference Voltage
- **RESET** - Microcontroller Reset
- **VIN** - External Power Input

Revisions

Version	Date	Modifications
1.1	February 7, 2019	- Added Revisions section. - Removed RX/TX use limitation for Arduino Uno WiFi Rev2 board. - Changed const from `int` to `byte` in LED example. - Added example for selectively compiling source code depending on which board is being used. - Made general wording and formatting changes for clarification.
1.0	December 7, 2018	Initial version.