DR. AL FISCHER

CHEM 191: A TEENSY INTRODUCTION TO ATMOSPHERIC CHEMISTRY

Preface

Getting Started with Arduino

Objective: Install Arduino and Teensyduino **To turn in:** Nothing! Make sure your software is installed and working so you're ready for the next class period.

Introduction: What is Arduino?

Arduino is an open-source platform for developing interactive electronic devices that began in 2005 as a student project at the Interaction Design Institute Ivrea in Ivrea, Italy. More specifically, Arduinos are *microcontrollers*, which are essentially small computers turn can be programmed to perform a specific task over and over again. Microcontrollers are ubiquitous in the modern world and are embedded in scientific equipment, cars, cell phones, and nearly every other electronic device. Arduino provides a simplified interface for microcontrollers and has become a very popular tool for hobby projects, citizen science tools, and even art installations.

Using an Arduino requires a *program*, or a set of commands that are uploaded to the Arduino to have it perform the desired task. In the Arduino community, the term *sketch* is used synonymously with program. Examples of possible programs include having the Arduino **measure** things like temperature, light, or humidity, or **control** things like lights or motors. Although the Arduino can run as a stand-alone device once setup, it must first be programmed by a computer.

To program the Arduino, we'll use the *integrated development environment* (IDE) available on the Arduino website. The IDE is shown in Figure 1 below. Arduino uses the computer language *C*–probably the most commonly used language in the world–but the IDE handles some of the programming behind the scenes to make it easier for the user. In this exercise, we'll use an Arduino to turn a light-emitting diode (LED) on and off, and you'll experience Arduino and the IDE firsthand.

Part of the beauty of Arduino is its open-source nature. Many spin-offs and flavors of Arduino exist due to it's open-source nature, each with its own unique benefits. In this class, we'll use a variant

of Arduino called the ***Teensy*** (specifically Teensy 3.5). The Teensy is programmed in the same way as an Arduino, but has a smaller form-factor and more capabilities than a standard Arduino.

This exercise will walk you through installation of the Arduino IDE, the software for the Teensy (Teensyduino), and an initial check of the board.

Install the Arduino IDE

You MUST follow the order and instructions here or Teensyduino will not work!

1. Use a web browser to navigate to `https://www.arduino.cc/en/Main/OldSoftwareReleases#previous`.
2. Download the installer for version **1.8.6**. You must choose the installer appropriate for your operating system.
3. Choose **Just Download** to download the software, or make a donation if you're feeling generous!
4. Open the file that downloads and run the installer.
5. Open the Arduino program to make sure it runs and to prepare for the next steps.

Install Teensyduino

1. Use a web browser to navigate to `https://www.pjrc.com/teensy/td_download.html`.
2. Download the correct installer for your operatring system. Note that steps 1 and 2 on the Teensy page should have been completed when you installed the Arduino IDE, above.
3. Run the installer. When prompted:
4. Select the `Arduino/` folder for the install location. If **Next** is greyed out you probably downloaded the wrong version of the Arduino IDE.
5. Select **All** when asked which additional libraries to install.

Test the Teensy!

Check the board

1. Plug your Teensy into the computer via the USB connection.

   Be ***EXTREMELY*** careful with the USB connection on the Teensy. They break ***VERY*** easily!!

1. You should see the orange LED blinking on the Teensy. If you do not, something is wrong with your Teensy and you should notify your instructor.

Check the software

1. Open the Arduino IDE.
2. Click `Tools > Board > Teensy 3.5`
3. Click `File > Examples > 01.Basics > Blink`.
4. Find the part in the program that says `delay(1000)` and change it to `delay(500)`.
5. Click **Verify** in the software (checkmark button in upper left corner).
6. Press the white button on the top of the Teensy.
7. Press upload in the software (right arrow button, next to checkmark).
8. You should see the orange LED blink more quickly. If you encountered any errors, something could be wrong with your software installation or you may have made a typo in the code. Notify your instructor of any problems.

Blink an LED

Objective: Use Arduino to turn an LED on and off. **To turn in:** You working Arduino code at the end of the exercise.

Introduction

This exercise will explore two concepts: (1) connecting devices to the Arduino and (2) the structure of the Arduino program used to control those devices. As a first example, we'll connect an LED (light-emitting diode) to the Teensy and write a program to turn in on and off.

How are things connected to a Teensy?

The Teensy can control all sorts of devices, including lights, motors, and various sensors. The device to be controlled must be connected to one of the Teensy's *pins*. You can think of a pin as a unique port on the Teensy. Pins can send and receive information, so communication between the pin and the sensor goes both ways. One pin might be used to control a light, while another might be used to read data from a temperature sensor. The pins that are available on the Teensy are shown in the figure below. Some pins provide power (3.3 V), some pins provide a connection to the circuit ground (GND), and some function to send and receive data. Some pins may have several function, each represented by a different color rectangle next to the pin the figure below. Note, though, that each pin can only have one function at any given time. For now, we'll just focus on the pin numbers. The labeled 0-33 (shown in grey) are called *digital pins*; they function as both inputs and outputs, but we'll use just the output functionality in this exercise.

There are several ways to connect something to the pins of the Teensy. One way is to directly solder items to the pins of the Teensy, in which a soft metal alloy is melted onto two parts as they are connected to make a permanent, metal connection. This is best saved for a final, fully vetted design that won't ever need to be changed.

For this exercise, we'll use another method, called a solderless breadboard. A breadboard is shown in the figure below. The holes in the breadboard are spaced in such a way that a Teensy can be plugged directly into the board. Columns, labelled A, B, C, . . . , are connected across the board electrically, such that A1 and B1 are connected; rows are insulated, such that A1 and A2 are not connected. The connections labeled with red and blue lines on each edge are connected together and are sometimes called rails; they provide an easy way to distribute power and ground connections around the board. If that seems confusing, try watching this video this video.

What is a program?

A *program*, or *sketch* in Arduino jargon, is a set of commands stored on the Teensy that tell it what to do. Although a Teensy can do most anything a computer would do, it can only run a single program over and over. For example, this exercise will use a set of commands to program the Teensy to turn an LED on and off. Other options would be to collect data from a temperature sensor, or turn a motor on a off to move a robot (or to do all of those things at once).

An Arduino program always consists of two parts: (1) a `setup()` function and (2) a `loop()` function. The `setup()` function runs once every time the Teensy is powered on. After that, the Teensy runs the `loop()` function over and over until it's turned off. Often, some initial lines of code are included above the `setup()` function that define things used in the program.

In the exercise, we'll connect an 3-color LED to a teensy and write a sketch to control it.

Connecting the LED

1. Push your Teensy into the breadboard provided and connect the USB cable.

   Remember, be ***EXTREMELY*** careful with the USB connection on the Teensy. They break ***VERY*** easily!!

2. Push the LED into the breadboard such that none of the leads (wires) are connected to anything else (each one is in it's own row). Make sure you note which row the longer lead goes into.

   Always work with the Teensy unplugged from the computer. This will help ensure no magic smoke escapes.

3. Use a jumper wire to connect the **longer lead** to the **GND** pin on the Teensy.

4. Use 3 more jumper wires to connect the remaining three leads to pins **13, 14, and 15**.

Program the Arduino

One Color

1. Open the Arduino IDE and load the Blink example (`File > Examples > 0.1 Basic > Blink`)
2. Go to `File > Save As` to save the example under a new name in the default location (`Documents/Arduino`). Call it **lastnameFirstname_blink.ino**.

   Remember, files that don't follow the naming convention *exactly* will incur point deductions.

3. You should see the following at the top of the sketch:

```
// Pin 13 has the LED on Teensy 3.0
// give it a name:
int led = 13;
```

Anything following a `//` is a comment; it does not affect the Arduino program. Please get in the habit of using comments to describe what each line of code does.

The line `int led = 13` defines the name of pin 13 as `led`. This name could be anything you wish. In this case, led is useful because the LED is connected to pin 13.

4. Next is the `setup()` function. In this case, the only setup necessary is to set the mode of the `led` pin as an output.

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

5. Finally, the `loop()` function, which runs over and over.

```
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
```

6. Assuming all those sections look good, press the **Upload** button (right arrow in top left).

Three Colors

You should have seen a single color blinking on and off. We really want all three to go on and off. Let's make some modifications to the code so that will happen.

1. Define the two extra pins. In this case, it could be useful to refer to them by color.

```
int blue = 13;   // blue pin connected to pin 13
int green = 14;  // green pin connected to pin 14
int red = 15;    // red pin connected to pin 15
```

2. Set each of those to outputs in the `setup()` function. You will have to fill in the blanks for the red and green pins below.

```
// the setup routine runs once when you press reset:
void setup() {
  pinMode(blue, OUTPUT);     // initialize the blue pin as an output.
  pinMode(____, OUTPUT);     // initialize the green pin as an output.
  ____________________       // initialize the red pin as an output.
}
```

3. Finally, add the extra colors to the `loop()` function.

```
// blue
digitalWrite(blue, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1000);                // wait for a second
digitalWrite(blue, LOW);    // turn the LED off by making the voltage LOW
delay(1000);                // wait for a second

// green
digitalWrite(green, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1000);                 // wait for a second
digitalWrite(green, LOW);    // turn the LED off by making the voltage LOW
delay(1000);                 // wait for a second

// red
digitalWrite(red, HIGH);     // turn the LED on (HIGH is the voltage level)
delay(1000);                 // wait for a second
digitalWrite(red, LOW);      // turn the LED off by making the voltage LOW
delay(1000);                 // wait for a second
```

4. Make sure you show your code and blinking LED to your instructor before you move on.

Make Your Own Changes

1. Before you leave, modify your code to change the blink pattern. You can change the sequence of the colors, the time between blinks, the number of blinks, or the amount of time each color is on.
2. Save your final code and turn it in to Dropbox.

Voltage

Objective: To turn in:

Functions

Objective: To turn in:

Soldering and PCB Assembly

Objective: To turn in:

Testing and Calibration

Objective: To turn in:

Appendix A: Arduino Cookbook

Recipe 1

Blink

Appendix B: Gory Details

Using the Snooze Library

The Snooze library for Teensy allows us to put the Teensy into a hibernation mode, wherein it uses very little power. The Snooze library and it's documentation can be found here: https://github.com/duff2013/Snooze.

The library allows for numerous wakeups, including checking if a button is pressed, using a delay, and checking for a voltage threshold to be crossed among other options. We'll use the real-time clock (RTC) wakeup for this project. The RTC wakeup allows the user to wake the Teensy up after a defined amount of time, as counted in hours, minutes, and/or seconds by the RTC on the Teensy.

The first step to using Snooze is to call the library, load the alarm driver, and configure the SnoozeBlock to use the alarm driver. All of this goes *before* the `setup()` portion of the program.

```
#include <Snooze.h>
SnoozeAlarm alarm;
SnoozeBlock config_teensy35(alarm)
```

Then, we set the RTC timer for the desired snooze period (in the `setup()` function). In the example below the Teensy will wake up after 5 minutes.

```
alarm.setRtcTimer(0, 5, 0); // h, m, s
```

Finally, we put the Teensy into hibernation mode `loop()` function. Other options exist, such as deep sleep and sleep, but hibernate uses the least power.

```
int who;
who = Snooze.hibernate(config_teensy35);
```

The Snooze function will return the pin number that caused the Teensy to wake up. The RTC wakes the Teensy up on pin 35. To know when that's happened, we just wait until `who` is set to 35.

```
if (who == 35) {
  # do something here
}
```

Using the RTC

- The Teensy 3.5 has an oscillator onboard. The battery is included on the WheeCAIR PCB.
- The coin cell (CR 1225) battery allows the RTC to keep time even when the board is not plugged in.
- The RTC is not temperature compensated so it may drift over time.
- The RTC runs during hibernation at full functionality. However, the Time library needs to resync everytime the Teensy wakes up using `setSyncProvider(getTeensy3Time);`

Char[] vs. String

Appendix C: Functions

getTeensy3Time()

This function comes from the RTC example.

```
time_t getTeensy3Time() {
  return Teensy3Clock.get();
}
```

setBMESamplingParameters()

```
void setBMESamplingParameters() {
  if (!bme.begin(0x76)) {
    return;
  }
  // Set up oversampling and filter initialization
  bme.setTemperatureOversampling(BME680_OS_8X);
  bme.setHumidityOversampling(BME680_OS_2X);
  bme.setPressureOversampling(BME680_OS_4X);
  bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
  bme.setGasHeater(320, 150); // 320*C for 150 ms
}
```

initializeBoard()

```
void initializeBoard() {
  pinMode(33, OUTPUT);
  digitalWrite(33, HIGH);
  setSyncProvider(getTeensy3Time);
  if (!SD.begin(chipSelect)) {
    return;
  }
  if (!card.init(SPI_HALF_SPEED, chipSelect)) {
        // don't do anything more:
        while (1) {
```

```
            blink();
            blink();
        }
    }
}
```

pmStartMeasurement()

```
bool pmStartMeasurement() {
    Serial1.begin(9600);
    delay(500);
    bool pmStatus = my_hpm.stop_autosend();
    pmStatus = my_hpm.start_measurement();
}
```

pmStopMeasurement()

```
bool pmStopMeasurement() {
    bool pmStatus = my_hpm.stop_measurement();
    delay(500);
    Serial1.end();
}
```

writeFile(char filename[16])

```
void writeFile(char filename[16]) {
    File dataFile = SD.open(filename, FILE_WRITE);
    if (dataFile) {
        dataFile.print(printDateTime(printDate(year(), month(), day()), printTime(hour(), minute(), second
        dataFile.print(printData()); dataFile.print('\t');
        dataFile.print(p25); dataFile.print('\t'); dataFile.println(p10);
        delay(500);
        dataFile.close();
     }
     else return;
}
```

blink()

```
void blink() {
    digitalWriteFast(LED_BUILTIN, HIGH);
    delay(15);
    digitalWriteFast(LED_BUILTIN, LOW);
```

```
    delay(15);
}
```

fillDigits(int digits)

```
String fillDigits(int digits) {
  if (digits < 10) {
    String digitsout = "0" + String(digits);
    return digitsout;
  }
  else return digits;
}
```

printTime(int h, int m, int s)

```
String printTime(int h, int m, int s) {
  return fillDigits(h) + ":" + fillDigits(m) + ":" + fillDigits(s);
}
```

printDate(int y, int m, int d)

```
String printDate(int y, int m, int d) {
  return String(y) + "-" + fillDigits(m) + "-" + fillDigits(d);
}
```

printDateTime(String d, String t)

```
String printDateTime(String d, String t) {
  return d + ' ' + t + '\t';
}
```

printData()

```
String printData() {
  return String(bme.temperature) + '\t' + String(bme.pressure/100.0) + '\t' + String(bme.humidity) + '\t'
}
```

createFileName(int y, int m, int d)

```
String createFileName(int y, int m, int d) {
  return String(fillDigits(y)) + String(fillDigits(m)) + String(fillDigits(d)) + deviceID + ".txt";
}
```

Appendix D: Strategies for Troubleshooting