# ME102 Lab 1: Getting Started

*This lab will teach you how to use the Teensy microcontroller in conjunction with the Arduino IDE (Integrated Development Environment). These instructions have been tested for the computers available in lab. However, it is just as easy to setup the IDE on your own laptop. You may wish to do this for future lab assignments.*

1. Introduction to the Teensy
    - Kit contents check
    - The Teensy and some terminology
    - Hello World
    - Anatomy of a sketch
2. Setting up your hardware buffers
    - Digital input buffering with the 74LS14
    - Digital output buffering with the 7417
    - Analog to Digital Converter (ADC) setup with the 4342
3. Clean up and check off
    - Blinking LED with the buffered setup

## 1| Introduction to the Teensy and the Arduino IDE

### Parts List

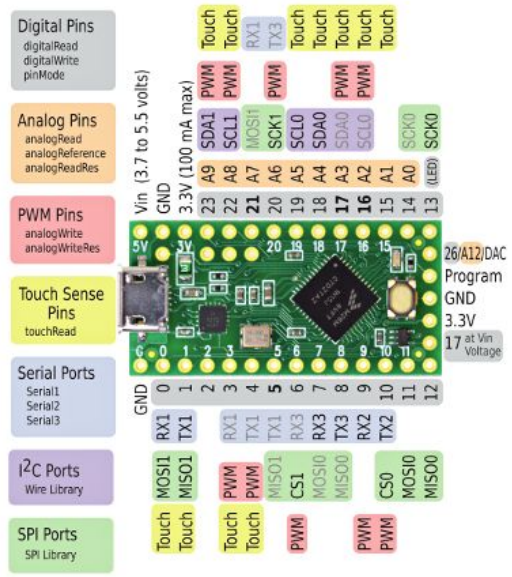
The following parts should be provided at your lab station:

- 1 Teensy 3.2 Microcontroller
- 1 Micro USB to USB Cable
- 1 Breadboard
- 1 Pair of Wire Cutters
- 1 DM7417
- 1 74LS14
- 1 TLV2374I

If you are missing any parts, please find Tom Clark.

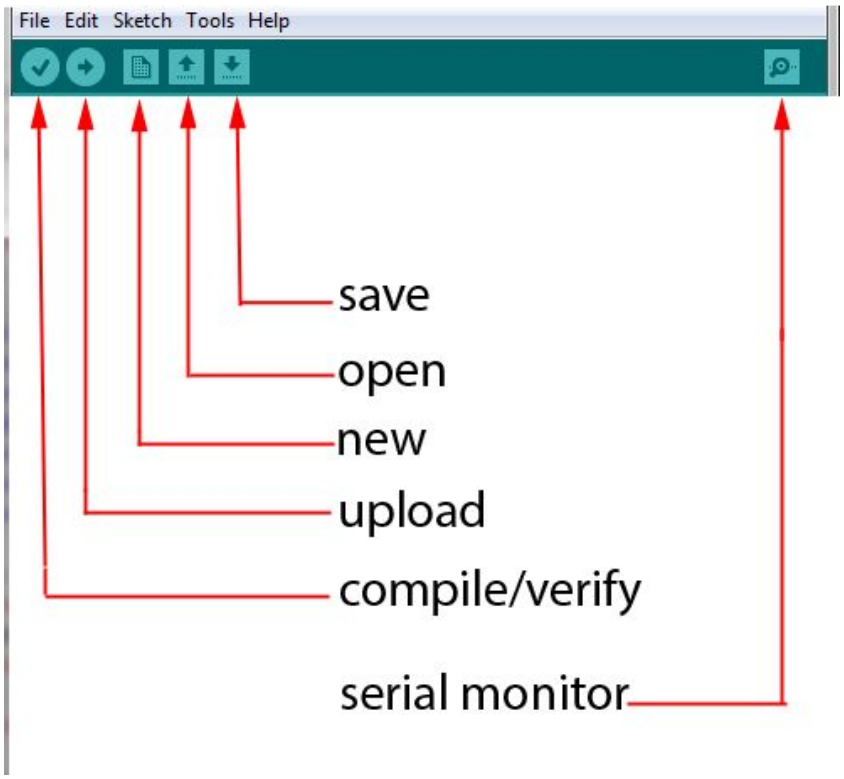
## The Teensy and some terminology

Your Teensy board will look something like this




Instructions on how to install a copy of the Arduino IDE on your own machine can be found here (Mac, Windows and Linux).

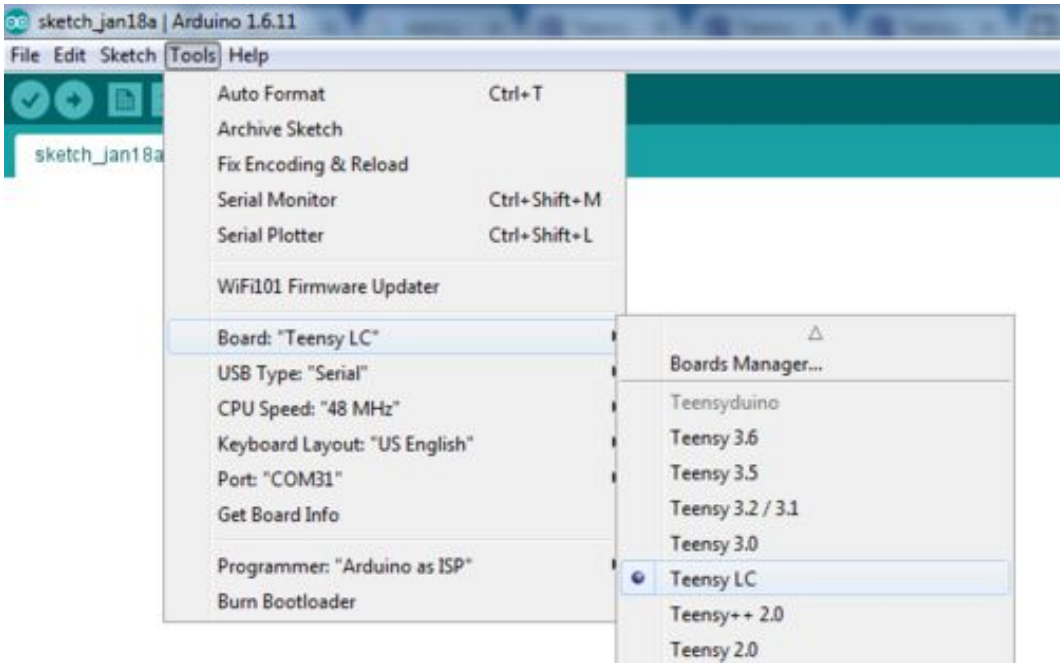
1) Start the Arduino IDE by clicking the Arduino icon on the desktop.
2) Specify the board you are using under "Tools -> Board".
3) Select the Teensy COM port under "Tools - > Port". **********



There are 7 shortcut commands that you can use with the Arduino IDE:

- **Serial Monitor** opens the only debugging tool you have with the Arduino. The Serial Monitor displays information passed from the Teensy to the computer.
- **Upload** compiles your program ("sketch" in the parlance of the creators of the Arduino), and sends it to the board if there are no compile-time errors.
- **Save** your sketch
- **Open** an existing sketch
- **New** creates a new sketch
- **Compile** your sketch in order to check for compile-time errors. As your sketch grows in size, it'll save time to verify that your code compiles before trying to upload it.

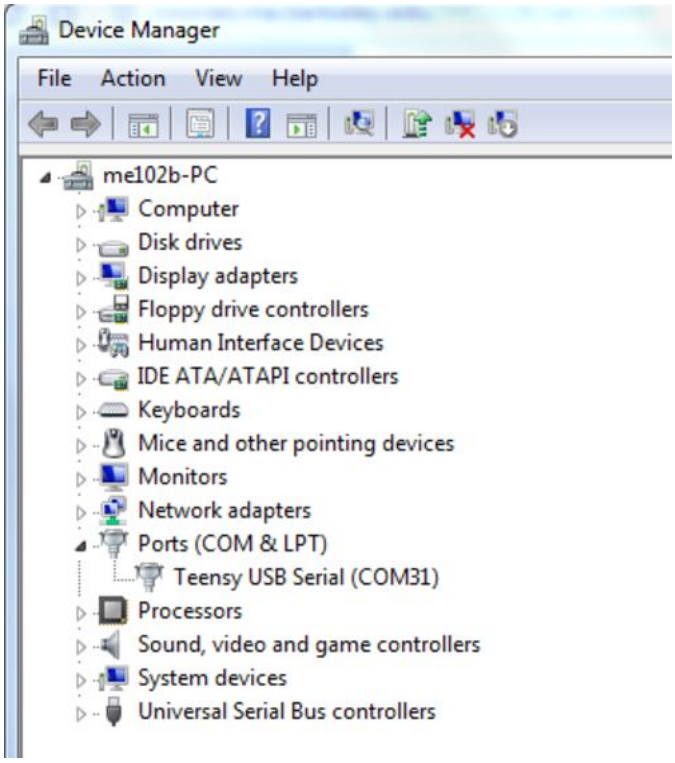
There are only a few things you need to configure and to learn before you begin. First, you need to make sure that the IDE has the correct target chosen. Select the Teensy LC from Tools>Board.



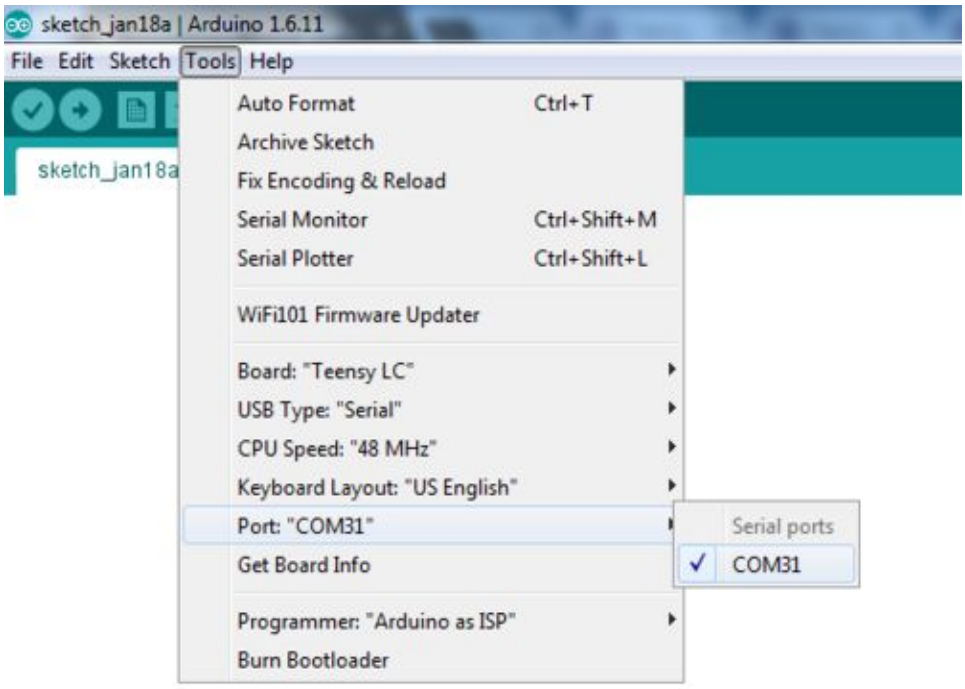

Secondly, the board communicates with the computer via a USB interface, but with a serial protocol. All this means is that you must select the correct serial port number. First, plug the provided Micro USB to USB cord into the computer. Once it is complete, open up the Device Manager (select Start and search for "Device Manager") and check for the COM port that says "Teensy USB Serial".

Sometimes the Teensy won't show up on the Arduino IDE. Try these things to get it to connect.

1) Look for the "Installing Hardware icon at the bottom corner of your screen (Windows). If you see it still hourglassing you just have to wait. Sometimes it can take a while.
2) Pressing the Load button on Teensy.
3) Click the upload button on the Arduino IDE.
4) Unplug and replug the USB cable
5) Open up the Device Manager (select Start and search for "Device Manager") and check for the COM port that says "Teensy USB Serial"




Make sure that the same COM port number is checked in the Arduino IDE.




HINT: Before you plug in the board, check the Serial Ports listed. Once you plug the board in, a new serial port should show up after 30 sec - 1 min. If you don’t see a new COM port, most likely a driver isn't installed correctly for that board. In this case, try restarting your computer.

## Hello World

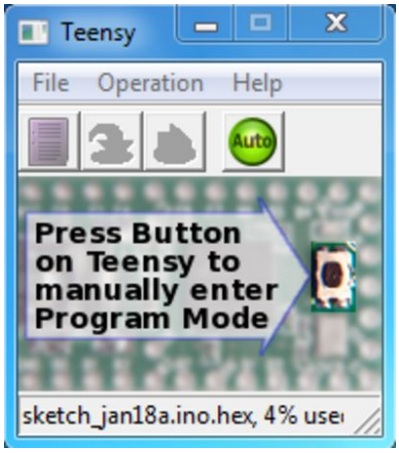
There are two mandatory functions in every sketch - setup() and loop(). As their names imply, setup() is where you will choose how to use the board, and loop() can be thought of as a never-ending while-loop. Setup() is executed once; time loop() repeats until the board is powered down.

First, copy and paste the following code into your new sketch in the Arduino IDE:

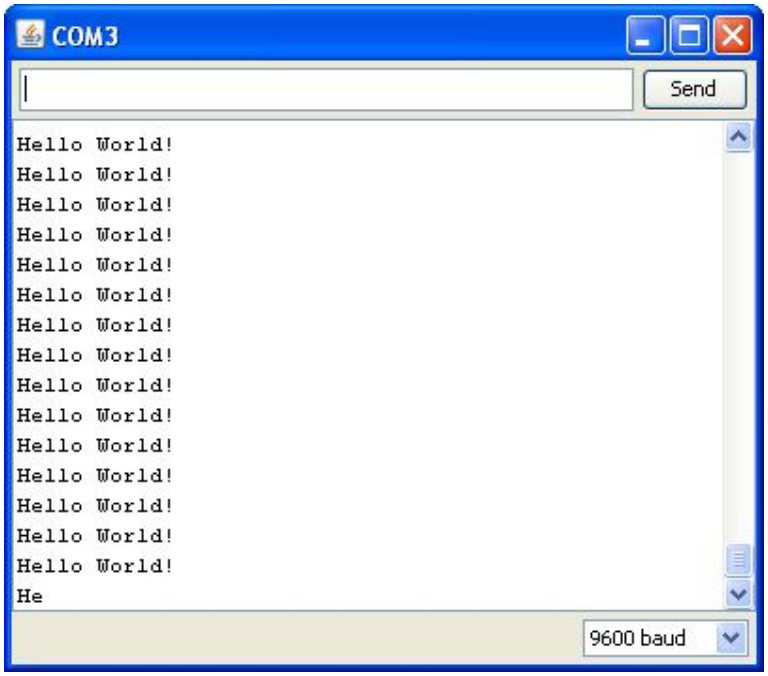
```
void setup() {
Serial.begin(38400);
}
void loop() {
Serial.println("Hello World!");
delay(1000);
}
```

Save this sketch to your desktop as "Hello World".

Next, compile and upload this code to the Teensy. Every time you upload code to the Teensy, you will see a box pop-up on your screen. This box will look like:




The first time you upload code to the Teensy, you need to press the button on your Teensy as shown in the image above. This allows the Teensy to enter into "Program Mode". You should not have to press this button for future uploads.
In this sketch, the need for serial communication is declared in the setup function. Inside the setup function, "Serial.begin(38400)" initializes the communication (the number 38400 is the baud rate in bits/s). This number is not important to us right now. Then, using the command "Serial.println" the board constantly returns "Hello World!" every second (due to the 1000ms delay) via the USB serial port.
Now, open the Serial Monitor. You should see this:




SUCCESS!!!

Congratulations, you've successfully compiled, uploaded and communicated serially with the Teensy.
Now lets try blinking the onboard LED. Copy and paste the following code into a new sketch named "Blink":

```
int ledPin =  13;    // LED connected to digital pin 13
void setup()   {
pinMode(ledPin, OUTPUT);
}
void loop()
{
digitalWrite(ledPin, HIGH);
delay(1000);
digitalWrite(ledPin, LOW);
delay(1000);
}
```

In setup(), pin 13 is declared to be an output. Then in loop(), the command "digitalWrite" sets pin 13 to logical HIGH, and nothing else is done as we wait for a delay of 1000ms. After which, pin 13 is set to logical LOW, and turned off. This behavior is then repeated ad nauseum. You should see the onboard LED next to the digital pin 13 blinking.
The next part of the lab focuses on setting up buffers to protect your board from mistakes. Although the Teensy is relatively cheap, not all microprocessors are.

## 2| Setting up the hardware environment

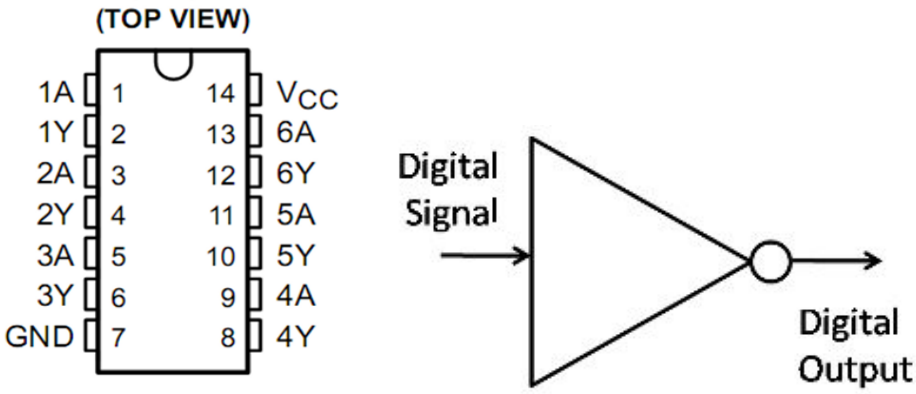
While the sensors and outputs in the labs generate and draw little current, that is not the case for everything you will encounter in the course of the ME102B project. Buffering the inputs to and outputs from the Teensy adds a layer of protection. If an accidental short suddenly appears, the cheap buffer will break, and the board will be preserved.

### Powering the breadboard

1. Supply 5V to the rails of the breadboard from the power supply provided in lab.
2. Make sure that the two rails on your breadboard share a common ground with whichever power supply you choose.

## Digital input buffering with the 74LS14

To buffer the digital input of the Teensy, we use the **74LS14** (datasheet). For this lab, we are not inputting any signal to the Teensy. However, understanding digital input buffering is potentially useful for your projects. The 74LS14 is a hex schmidtt trigger inverter. It takes a digital input and outputs the opposite. HIGH becomes LOW, and LOW becomes HIGH. In TTL (Transistor-to-Transistor Logic), a HIGH signal is anything from 2.2V to 5V whereas a LOW signal is anything from 0V to .8V. The area from .8V to 2.2V is essentially undefined behavior. The schmidtt trigger is a property of this particular chip that allows it to be slightly less responsive to noise. And finally, there are six such inverters on a single chip (hence hex).
To setup the 74LS14 :

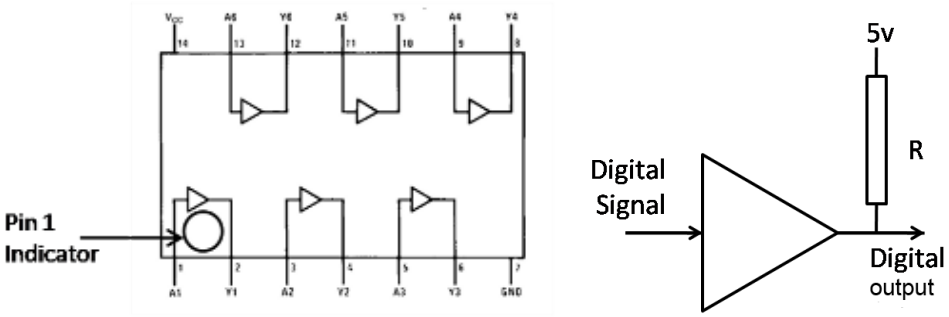



1. Connect the Vcc to the 5V rail and GND to ground
2. Verify that the installed IC behaves correctly. For example, you could use the oscilloscope to monitor both input from a function generator and the inverter's output. Using a square wave is the easiest, but you could use anything signal from the function generator as long as the minimum voltage is 0 and maximum voltage is 5.

**Function Generator Note:**
A digital signal is a binary signal which is either LOW and HIGH. For the Teensy, LOW is 0V/GND and HIGH is 5V (it is quite common for logic high to be 3.3V for other circuits). To get a signal from 0V to 5V using the function generator, a square wave with a low signal at 0V and a high at 5V is required. Set an amplitude of 2.5V, but due to the way the function generator is designed, an offset of 1.25V gives an overall signal from 0V to 5 V.
The Function generator is used for testing radio circuits, and is anticipating a resistance of 50 Ohm at the output. Due to maximizing power consumption the internal resistance of the function generator is about 50Ohm. However the resistance of the oscilloscope is ~inf. This causes the output of the Function generator to be ~double.

## Digital output buffering with the DM7417

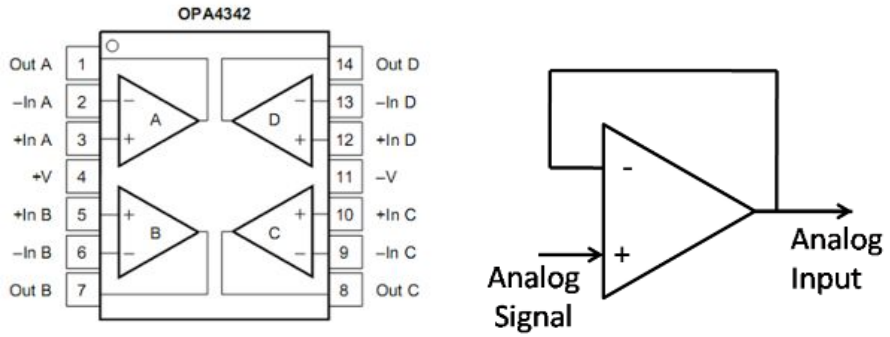
Digital output buffering with the **DM7417** (datasheet) protects the Teensy. The DM7417 is a hex buffer with open collector high voltage output. An open-collector only guarantees that a LOW input results in a LOW output. There are no guarantees for a HIGH input.
By using a pull-up resistor on the output end, one can supply any level of voltage within the operational specifications of the IC. For the DM7417, it is 15V.
The basic function of a pull-up resistor is to insure that given no other input, a circuit assumes a default value. Consider the circuit in "Configuration of DM7417", when the digital signal input is HIGH, the IC acts as a high impedance part so current flows to the right. When the digital signal input is LOW, the IC becomes a current sink. In order to prevent IC destruction, **pull-up resistors are generally in the range of 10k to 47k ohms.** However, there are special cases. The pull-up resistor limits current flow. With a 10k resistor, current flow through it is limited to .5mA. This is not enough to drive an LED. Given that the SN7417 can sink 40mA, **use a resistor of around 1k ohms as your pull-up resistor.**




1. Connect the Vcc to the 5V rail and GND to ground
2. Connect a 1k Ω pull-up resistor (or something close to this resistance) between one of the IC outputs and the 5V rail
3. Verify that the installed IC behaves correctly. For example, you could use the oscilloscope to monitor both input from a function generator and the buffer's output. It will be pretty boring since the buffer's output value is its input value if a square wave is used.

### Analog to Digital Converter (ADC) setup with the 4342

If we had an analog input to our Teensy, we would protect the ADC with a voltage follower. Again, we have no analog input to our Teensy right now. However, this is also good to know for your projects. In your lab kits, you may either have a OPA4342(datasheet) or a TLV2374I(datasheet). Both are suitable for our purposes.
To setup the voltage follower with the op amp:




1. Connect the +V pin to 5V and the –V pin to ground
2. Connect the inverting input (-In C) to the output (Out C) for any one of the four op amps on the IC. This creates the voltage follower.
3. Verify that there is unity gain or Vin = Vout for your voltage follower. You can easily do this by connecting the wiper of your potentiometer, which has been connected between 5V and ground, to the non-inverting input (+ In C) . The wiper is the part that changes voltage as the knob turns.

## 3| Clean up and check off

### Blinking LED with the buffered setup

The check off for this lab is a blinking LED that works with your digital output buffer.

1. Connect Pin13 to the one of the 7417's inputs
2. Connect the corresponding output from the 7417 to the anode of the LED.
3. Note that the onboard LED and your recently connected LED should blink in phase.