School of Advanced Technology

CST8227 Lab 2: Up and Running with some basic sketches

Lab Objectives:

1. Set up the protoboard with the microcontroller.
2. Run a simple sketch which serves as a "Hello World" app.
3. Write a few custom apps using digitalWrite().
4. Set up a LED circuit and verify operation.
5. Verify operation of a pushbutton input using digitalRead();

Required Equipment:

- Laptop with sufficient space available (e.g. several GB)
- A working network connection
- A standard USB 5-pin Micro-B cable supplied **by you.**
- An Arduino compatible Teensyduino 3.2 Microcontroller
- Suitable length and color connection wires
- A red light emitting diode
- A pushbutton switch with a suitable size resistor
- A 220 (or 180) Ω resistor

Part A: Hardware and Software Setup

1. Verify that the version of board you have is a Teensyduino 3.2, based on the silk screen markings found somewhere on the board.
2. Mounting the microcontroller in the protoboard and first operation. Lay out the circuit as shown in graphic #1 (uploaded). Make certain that the USB connector end of the teensy microcontroller is installed starting at the top row of through holes. Wire the ground (GND) and power rails. The microcontroller and GND/power rails layout will remain permanent for the duration of the course. Do not uninstall the teensy microcontroller from the protoboard, when you are finished experimenting with it. It is less likely to get damaged.
3. Carefully connect the Teensyduino board to your computer using the USB micro-B cable. After some drivers are installed (automatically when your PC attempts to recognize the new device), you should see the onboard LED start to blink on and off (it comes preloaded with both the bootloader program and the sketch Blink.ino.
4. Open the Arduino IDE and explore the menu bar and explore and Configure the Arduino IDE. In order to correctly communicate with the Teensyduino board, the Arduino IDE needs to know the specific model of board being used. Set this parameter from within the Arduino IDE by accessing the **Tools -> Board** menu. Select your model in the list of boards presented (see http://www.pjrc.com/teensy/td_usage.html). You'll need to make sure this is done correctly, particularly because the Teensy is an Arduino-*compatible* board rather than a brand-name Arduino. In the **Tools -> USB: Type** menu, select **Serial**. This will set the desired behavior of the board's USB interface. Choose a CPU Speed of 72 MHz. Remember though: faster chip = more power consumed.

Part B: Writing and uploading a custom sketch: change the frequency of "Blink" [demo #1]

1. Open the Blink.ino sketch. You may find this sketch by accessing **File -> Examples -> Teensy -> Tutorial 1 -> Blink**. Alternatively you can write a completely new sketch by opening a new file using the **File -> New** drop down menu.
2. Examine the code in the Blink.ino sketch. For now you will only be editing the argument of the delay() function. Teensyduino sketches require two methods: setup() and loop(). The setup() function is for utility type functions such as specifying whether a pin is to be an input or an output when using the pinMode() function. The setup() function is run once, at the beginning of the app. The loop() function runs infinitely. Define any constants before both the setup() and loop() functions. Like all good programming practices, associate meaningful variable names with pin numbers (example: int ledPin = 13).
3. In order to change the duration of the ON/OFF cycles, you will need to experiment with the argument for the delay() function.
4. It is a good policy to compile your code before uploading it, on account that technically, you could upload a "bad" piece of code. This could yield unexpected results or even damage the chip. Compile by clicking the "**Verify**" button on the task bar. If you get any errors, go back and fix them until the program compiles successfully. Look for the status message indicating that the compilation was successful.
5. Upload and demo.

Part C: Using a discrete light emitting diode (LED) [demo #2]

1. Instead of using the onboard LED at pin 13, now install a LED in series with a resistor, and connect the circuit to a Teensyduino pin. Refer to the lab2PartD.bmp file for a suggestion regarding how to lay this circuit out. For now, use a red LED in series with a 220 (or 180) Ω resistor. The LED has polarity and must be installed correctly – the longer lead of the LED is connected to the Teensyduino side of the circuit and the shorter lead of the LED to the GND part of the circuit. Have your circuit checked before you apply power. Run the same "Blink" sketch except of course, with the LED pin number changed.

Part D: Using the pushbutton switch with digitalRead() to control the LED [demo #3]

1. Install the pusbutton circuit in either the "pull-down" or the "pull-up" configuration. The output of the pushbutton circuit will be connected to a teensy pin. New lines of code in your sketch will include a new variable to hold the state of the pushbutton. This teensy pin that is connected to the pushbutton circuit receives input from the pushbutton circuit, and therefore must be configured to be an INPUT when specifying the mode of the pin in the setup() function.
2. Demo the LED toggling between ON and OFF whenever the user presses the pushbutton.

Troubleshooting Tips:

You'll need to get the hang of uploading code and resetting your Teensyduino board. It's generally automatic, but there may be circumstances when the relative timing of uploading & pressing the reset button makes a difference. Read the instructions and tutorial at http://www.pjrc.com/teensy/td_usage.html.

Demonstrations:

1. The *modified* blink.ino sketch using the onboard LED [demo #1].
2. Blink.ino running using the discrete LED [demo #2].
3. A pushbutton circuit using digitalRead() [demo #3].

Questions to Consider:

- What happens to the loaded app if you unplug and then plug the board back in?
- What happens to the loaded app if you press (and release) the reset button on the board?
- How many apps can you have loaded on the board at once?
- Where can you find the compiled program's size in bytes?
- What is the standard processor speed of a Teensy? (Not overclocked!)
- How much flash (program storage) is available on a Teensy?
- How much RAM is available on a Teensy?
- How much EEPROM (user writable) on a Teensy?
- How big (ie. how many bytes) is an "int" in the Arduino environment?
- How many different types of floating point data types are there on in the Arduino environment?

Challenge: What is the maximum number of int's that can possibly exist simultaneously?
Challenge: How big is the stack (ie. maximum size in bytes)?