6.101 Heartbeat Display

Massachusetts Institute of Technology
Lecturers: Gim Hom, Negar Reiskarimian
Credit: Henry Love[1]

Summary

This document provides instructions on how to connect and use the MSP1443 SPI TFT Module with a Teensy 3.2 to display beats-per-minute (BPM) of a periodic pulse. To test the setup, a normally-open, momentary push button will be used to provide the pulse to the Teensy. It will be your job to replace the pulse generated by the push button with your heartbeat.

1 Links

Hardware	
Teensy 3.2:	https://www.pjrc.com/teensy/card7a_rev1.pdf
MSP1443 SPI TFT Module:	http://www.lcdwiki.com/res/MSP1443/1.44inch_SPI_Module_MSP1443_User_Manual_EN.pdf
Software	
Arduino IDE:	https://www.arduino.cc/en/main/software
Teensyduino:	https://www.pjrc.com/teensy/td_download.html
6.101 Arduino Sketch:	https://github.com/hlove18/6.101-Heartbeat-Display

2 Setup

Depending on how much of this software you already have, all/some/none of these steps may be necessary. If you've programmed an Arduino before on your computer, it is likely you already have the Arduino software installed. If you've programmed a Teensy using the Arduino IDE from your computer, then it is also likely that you also have the Teesyduino software and the required libraries.

2.1 Install Arduino Software

Using the link above, install the Arduino Integrated Development Environment (IDE). This is a complete environment that provides API's and a compiler so you can quickly get started developing and running code on your Arduino/Teensy. After clicking the link, follow the instructions for your specific operating system.

2.2 Install Teensyduino

Teensyduino is a software add-on for the Arduino and allows us to run sketches on the Teensy and Teensy++! It lets us upload code to the Teensy more or less the same way we would with an Arduino. Following the Teensyduino link provided above, install the software on your computer. After clicking the link, follow the instructions for your specific operating system.

[1] 6.101 TA, MIT SB'18, MENG'10

3 Wiring

The Teensy will receive power from the USB port when plugged into a computer. For debugging and initial setup, powering the Teensy through the USB port is the recommended method. To power the Teensy without a USB cable, attach a battery to the `Vin` pin of the Teensy. Please note that voltage applied to this pin must be between 3.6V and 6.0V for the Teensy to operate properly. **Do not use a 9V battery!** The Teensy has an on-board voltage regulator that will ensure that the output of the 3.3V pin is indeed 3.3V, whether plugged into your computer via USB or connected to a battery. The 3.3V pin on the Teensy should be connected to the red power rail on your breadboard, and `GND` of the Teensy should be connected to the blue power rail on your breadboard. This is common practice and will simplify much of the wiring.

Table 1: Teensy Connections

Teensy Pin	Connected to	Description
8	`RESET` (pin 4) on TFT display	LCD reset signal, low level reset
9	`A0` (pin 5) on TFT display	LCD register / data selection signal
10	`CS` (pin 3) on TFT display	LCD chip select signal, low level enable
11	`SDA` (pin 6) on TFT display	SPI bus write data signal
13	`SCK` (pin 7) on TFT display	SPI bus clock signal
14	Push button and pull-down resistor	When the push button is pressed, this pin should see 3.3V, and `GND` otherwise.

Pin `1` (`VCC`) of the TFT display is indicated by the box around the pin. The power pins of the TFT display (`VCC` and `GND`; pins `1` and `2`, respectively) should be connected to the power rails of the breadboard; in this way, the TFT display will be powered by a regulated 3.3V, provided by the Teensy. The `LED` pin (pin `8` of the TFT display) can be connected directly to the 3.3V power rail. This pin controls the backlight brightness of the display. If you are feeling ambitious, you can connect a potentiometer to this pin to provide any voltage between `GND` and 3.3V for adjustable brightness, or, for maximum fun, use a photodiode to sense the ambient light and adjust the brightness automatically.

4 Testing

After the wiring is complete, upload the provided Arduino Sketch to the Teensy. You can download the Arduino sketch from the link provided above. Please note that the sketch will have to be in a folder with the same name as the sketch itself. The Arduino software will prompt you with an message if this is not the case. Once you have the Arduino sketch downloaded and open in the IDE:

Go to Tools → Board and select Teensy 3.2 / 3.1

Go to Tools → Port and select the correct serial port. A quick way to figure this out is to unplug your USB cable and see which option disappears.

Click the “Upload” button (circle with an arrow in it) in the Arduino IDE to upload the provided sketch to the Teensy. This may take a few seconds, and you may have to press the reset button on the Teensy when the status of the IDE says “Uploading...”

If your code has syntax errors then Arduino will tell you in red text down at the bottom. Your code can still have bugs even if it has no syntax errors though. If you find a bug in the code, please tell Henry. If you believe the code is correct, then the mistake must be in the wiring. Use a meter in continuity mode to make sure that your circuit is connected in the way you expect. If all goes well, you should see a ”Welcome!” screen

after the sketch is uploaded to the Teensy successfully. After the welcome screen goes away, try pressing the button and see if BPM is displayed on the screen. After 4 seconds of inactivity, the screen will return to the ”Dead” state.

5 Extra Details

The code features a moving average filter which is a simple, low pass filter. The BPM you see displayed on the screen is not the BPM from just the last button press, but rather the average BPM calculated using the past 5 (default) BPM values. The number of samples that is used to calculate the average is called the ”window size” of the filter. If you would like to change the window size of your moving average filter, you can do so by assigning the `windowSize` variable to some other value in your code. Please note that the maximum window size is 20 and is determined by how much memory is allocated to the `pastBPM` array. This can also be changed, but you probably won’t need to do that.

There is a `debugging` variable in the code that lets you more easily observe the operation of the averaging filter. When `debugging` is set to `true`, information from the Teensy will be sent the the computer and displayed in the serial monitor. The serial monitor will display information such as the state of the code, the values in the `pastBPM` array, the sum of the array, the number of elements in the array (Index), and the calculated average BPM. To open the serial monitor, in the Arduino IDE, go to Tools→ Serial Monitor. The sketch must be running on the Teensy for you to see the output.

The code also features a debounce function that ensures that only one button press is registered when the button is pressed. In reality, due to the physical construction of the switch, a button press can cause the button to ”bounce” when it makes contact, appearing to the microcontroller as many button presses in quick succession to one another. We perceive a button press as one event, but the microcontroller (which operates must faster than us) can discern the small events unnoticeable to us. The debounce function waits 25mS until it registers another button press, enough time for the mechanical transients to settle.