Allison Okamura

Lab Assignment 4:
Hapkit Position Reading and Force Output

In this lab, you will start with the template code and modify it to:
- Print the Hapkit position to the screen
- Output a force to the Hapkit

The steps for this lab are provided below.

Step 1: Examine the Hapkit template code from last week

This code should still be in your mbed compiler window (www.mbed.com).

Examine the contents of several of the main C++ (cpp) files and their corresponding header (h) files:

- **main.cpp:** This is the main program file, which sets up an infinite loop to continuously run your code. Note that the Hapkit program is different from a typical program, which computes and then gives a result. The Hapkit program runs continuously, repeating the same commands over and over again in order to keep the haptic device running.

  Until we use the 2-dof Haplink, we will place our function calls under `#ifdef HAPKIT`

  Currently the functions running there are `calculatePositionHapkit()` and `blinkLED()`

  Also note that you can print to the serial line, e.g. `serial.printf("hello world\n")`

- **haplink_virtual_environments.cpp:** This file is where you will program force outputs based on measured positions. This is where the `blinkLED()` function is located, and you will write other functions here.

- **haplink_position.cpp:** Computes the Hapkit position and velocity from the encoder data. Do not change this file!

- **haplink_encoders.cpp:** Reads from the encoders. Do not change this file!

- **haplink_adc.cpp:** Reads from the analog-to-digital converters for the sesnor readings. Do not change this file (for now).

- **haplink_pwm.cpp:** Controls the output to the PWM channels connected to motors. Do not change this file!

Step 2: Check the position output

1. In **haplink_position.h**, examine the following definitions. Use a ruler to check these measurements and adjust slightly if needed.

```
#define R_A                 75.0      // radius of Sector A in mm
#define R_HA                70.0      // radius of Handle A for Hapkit in mm
#define R_MA                5.0       // radius of Motor A in mm
```

2. Print the position measurement (and some other things) by adding the following code to **main.cpp**:

   ```
   serial.printf("%lf, %lf, %lf, %lf,%lf\r\n",dutyPrint1,getXH(),getDxH(),
   ForceX, TorqueMotor1);
   ```

   Place this code underneath the "`hello world`" line, and comment out the "`hello world`" line by placing // at the start of that line.

3. After you have downloaded your program to the control board (with both USB cables and the 5V power plugged in), use the Arduino IDE's built-in **serial monitor** to view the result as you move the handle around. *Hint:* click on Tools > Serial Monitor in the Arudino IDE.

   What values change as you move the handle around? Does it make sense?

   Try checking or unchecking the box that says "Autoscroll" as you move, and see what happens.

Step 3: Output a (small) force

4. By editing **haplink_virtual_environments.h** (header file) and **haplink_virtual_environments.cpp** (C++ file), you will create a new function called `hapkitForce()`.

   Start by adding the function declaration to the header file. In the section identified by the comments

   ```
   /* Virtual Environments Functions*/
   //debug
   ```

   add a new function declaration as follows:

   ```
   void hapkitForce  ( void );
   ```

   Then add the function to the C++ file. In the section identified by the comment

   ```
   /***... Debug LED ...***/
   ```

   and after the `BlinkLED` function (which you should comment out), add a new function as follows:

   ```
   void hapkitForce( void )
   {
       ForceX = 0.5; // approximately 0.5 N
       TorqueMotor1 = -((R_MA/R_A) * R_HA) * ForceX;
       TorqueMotor1 = TorqueMotor1*0.001; //convert units
       outputTorqueMotor1(TorqueMotor1);
   }
   ```

5. Now call the new function from **main.cpp** by commenting out `blinkLED();` and adding `hapkitForce();`

6. Downloaded your program to the control board (with both USB cables and the 5V power plugged in), and use the Arduino IDE's built-in **serial monitor** to view the prints. *Hint:* click on Tools > Serial Monitor in the Arudino IDE.

   Before plugging in the 12 V power, make sure that the output on the serial monitor looks like:

   ```
   0.102200, 0.000000, 0.000000, 0.500000,-0.002333
   ```

   What do these numbers mean? Recall from the print function that we used earlier that these correspond to:

   ```
   dutyPrint1,getXH(),getDxH(), ForceX, TorqueMotor1
   ```

   Here is what they mean:

   `dutyPrint1` corresponds to the duty cycle (larger number is higher force)

   `getXH()` gives the position of the handle in mm (and it will still change as you move the handle)

   `getDxH()` corresponds to the velocity of the handle

   `ForceX` is the force you assigned (0.5 N in this case)

   `TorqueMotor1` is the calculated torque command to the motor calculated from `ForceX`

7. Now **while holding the handle to prevent it from moving**, plug in the 12 V power, which powers the motor. Get someone else at your table to help if you need an extra pair of hands! Once the motor is powered, does the Hapkit respond as you expected?

8. If you have time, try some other things by modifying the `hapkitForce` function in **haplink_virtual_environments.cpp**:

   - Lower the force from 0.5 to 0.1. Can you feel it? What is the smallest force you can feel?
   - Change the force to a negative value. The Hapkit should apply a force in the opposite direction.
   - Increase the force to 3.0. Hold on tight as soon as you are downloading, or the Hapkit will slam into the end of its workspace. You can make the force larger, but we don't advise it!
   - Use a small force (such as 0.5) and apply the force only if you are in the right side of the workspace. To do this, replace the line `ForceX = 0.5;` with the following two lines:

     ```
     if (xH > 0.0) ForceX = -0.5;
     else ForceX = 0.0;
     ```

     Does it behave as you would expect? Why do you think we made the force negative?