The goal of this assignment is to get used to using general purpose IO (GPIO). **Keep all your work in a github repository.**

## Buttons and LEDs

The goal here is to get some LEDs and buttons wired up and tested before doing the next part.

1. Wire up your breadboard to have 4 buttons. Each is to have one terminal attached to +3.3V and the other to a GPIO port.
2. Also wire up 4 LEDs with current limiting resistors. Tie the *plus* side of the LED to the GPIO port and run the *minus* to the resistor then to ground.
3. Write a simple program that reads the switches and lights a corresponding LED. Use interrupts. (callback in Python.)

## Measuring a gpio pin on an Oscilloscope

Get an oscilloscope so you can measure the output of your gpio pins. Run

bone$ **cd exercises/gpio**

bone$ **./togglegpio.sh 60 0.1**

and answer the following questions about gpio measurements.

1. What's the min and max voltage?
2. What period is it?
3. How close is it to 100ms?
4. Why do they differ?
5. Run **htop** and see how much processor you are using.
6. Try different values for the sleep time (2nd argument). What's the shortest period you can get? Make a table of the values you try and the corresponding period and processor usage.
7. How stable is the period?
8. Try launching something like **vi**. How stable is the period?
9. Try cleaning up togglegpio.sh and removing unneeded lines. Does it impact the period?
10. togglegpio uses bash (first line in file). Try using sh. Is the period shorter?
11. What's the shortest period you can get?

### Python

Write a python script to toggle a gpio pin. Answer the above questions for you Python script. Present the shell script and Python script results in a table for easy comparison.

### C

Repeat the above using C. Add your results to the table.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Bash | Python | C |
| 1 | -.06 to 3.42V | -.18v to 3.54v | -.18 to 3.62 |
| 2 | 236.9ms | 101.4 | 101.1 |
| 3 | The program takes in 100ms as half the period. Even with that it is till about 18.5% off. | Only 1.4% off | Only 1.1% off |
| 4 | This is most likely due to delays in file operations and basic bash overhead | Due to python overhead | Due to small over head of still having to write to the files through the libraries |
| 5 | 18% | 3.4% | 2.7% |
| 6 | .05: 134.4ms 29.1% | .05: 51.4ms 4.1% | .05: 51.1ms 2.8% |
| .01: 54.4ms 66.4% | .01: 11.5ms 14.2% | .01: 11.02ms 4.5% |
| .005: 45.5ms 80.7% | .005: 6.34ms 18.6% | .005: 6.02ms 2.5% |
| .001: 35.5ms 95.4% | .00005: 445ms 57.3% | .000005: 1.75ms 22.8% |
| 7 | At the lower periods it jumps around a little (+- 2%) | It jumps around a lot. A lot more than w/ the bash at its smallest period. However at similar periods, its more stable. | At its lowest period it fluctutates a little. Otherwise, its very stable |
| 8 | Opening programs makes the period jump a lot more but goes back to being stable if the program isn’t doing anything. Doing things in the program make it fluctuate again. | Opening programs barely affects the period at higher periods but has a rather drastic affect at very low periods | While it still fluctuates it’s a lot less than w/ the others |
| 9 | Yes, I was able to get the initial 236.5 down to 217.8 |  |  |
| 10 | Sh improves and gets the peiod down to 213ms |  |  |
| 11 | 12.6ms |  |  |

## Etch-a-sketch

Next write modify your Etch-a-sketch[1] program to be controlled by the pushbuttons. I suggest converting to Python since the I/O is easy. For now, just print the grid in the terminal window. Next week we’ll interface it to the LED grid.

0 1 2 3 4 5 6 7

0: x x x x

1: x x

2: x x x x

3: x x

4: x x x x

5: x x x x

6: x x

7: x x x x

My example shows an 8 by 8 grid. Make yours able to do any size.

Extras

Here are some other things you can do to make the project more interesting. Feel free to think of your own ideas.

* Add a button to clear the display, or toggle between writing and erasing.

### Modify togglegpio.c

1. Modify exercises/gpio/togglegpio.c to take the gpio number.
2. Presently one parameter specifies both the on and off time. Modify the code so the on and off times are controlled separately.
3. Add in interrupt handler (see gpio-int-test.c) to trap ctrl-C and close things properly.
4. What's the highest frequency you can see on the 'scope?

### Modify gpio-int-test.c

1. Modify exercises/gpio/gpio-init-test.c to count the number of times the User button has been pressed. Set **edge** to only count releases.
2. Copy gpio-init-test.c to **gpioThru.c** and modify it to copy the value of one port to another. You'll have to add code to open a second gpio port for writing (check **gpio\_set\_dir**).
3. gpio7 is attached to pin P9\_42 on the Bone. Attach a function generator to this pin. Be sure to set the HiLevel to 3.3V and the LoLevel to 0. Does the output track the input?
4. What's the highest frequency the output will track the input? What's the CPU usage?
5. What's the delay from when the input changes to when the output changes?

## What to turn in

1. Create a repository on <https://github.com/>

2. List your repository on the Embedded Repos link on Moodle.

3. Make a subdirectory in your github repository called **hw02**.

4. Put all your files in the directory, include a **ReadMe.md** that explains what your homework does and how to use it.

5. Document your code.

[1]<http://en.wikipedia.org/wiki/Etch_A_Sketch>