EECS 2031A 2018

Lab #8 GPIO pin output using BASH

EECS 2031 Lab #7

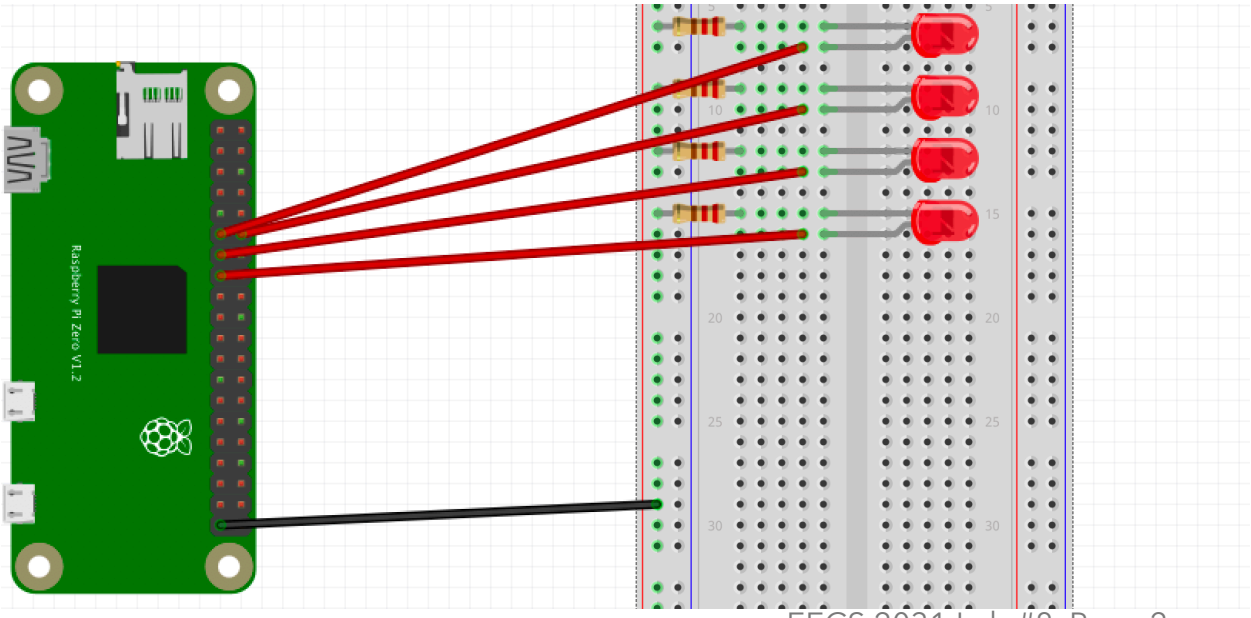
GPIO pin output using BASH

This lab deals with GPIO pin output using BASH. That is, to build a simple circuit with LED's making up a display and then using BASH to control power in the circuit. To make things easier for you, there is a common circuit that you will use throughout the lab although the lab will require you to write a number of different shell scripts to manipulate the circuit.

Four bit display

So we begin first with the circuit. In building this circuit the lab assumes that you have four working LED's at this point and four resistors. If you only have three, then only build a circuit with three LEDs. (If you have more, then lend one to someone who only has two.) But you will need at least three working LEDs to make this lab work.

The circuit you will use in this lab is shown below.



Basically, this is a collection of four LED's, each of which is connected via a resistor to ground and has power provided by WiringPi Pins 0, 1, 2 and 3. Physically the four LEDs should be laid out as shown so that they are in a row and wired up in order. This will make much more sense soon.

GPIO Command

Once you have the circuit built and are sure that it is wired correctly, power up your Raspberry Pi and log onto the machine. The gpio command can be used to control the gpio pins from the shell, much like you did using the wiringPi library in earlier labs. Full details on the gpio command can be found at wiringpi.com and also through the man command (man gpio) on the Raspberry Pi.

As with the wiringPi library, the gpio command can be used to manipulate the gpio pins on the Raspberry Pi. Critical commands include

- gpio mode pin out - sets the mode of 'pin' to be output
- gpio write pin value - sets the output value of 'pin' to be 'value'

There are, of course, many other possible arguments for the gpio command, and you are encouraged to test out a few of them. (man gpio is a very helpful place to start).

You are now going to develop a collection of shell scripts to manipulate the circuit you have created. You should create a GitHub site and clone this repository onto your Raspberry Pi. Include each of these scripts in your repository. Each script should start with an appropriate hash-bang line to ensure that the script is executed by /bin/bash. Each script should have its permissions set to 755 so that you can execute the file.

Bash scripts

initMode.sh - this should set the output mode of each of pins 0 through 3 inclusive to be output and set their current output value to be 0. (off). Note that gpio mode values remain in their input/output mode until changed, so once you run initMode.sh it should not have to be run again until you power the Raspberry Pi on and off. Note: Your code should use a for loop to loop over the pins, not just set each pin manually.

blink5.sh - This code should blink all four of your LEDs on and then off with 1 second of on time and 1 second of off time. It should do this five times.

setbits.sh - Imagine that the pins corresponds to bits in a 4 bit number (from 0 through 15). setbits.sh should take one argument, a number in the range 0 through 15 and set the on value of the four LEDs so that the bit pattern corresponds to the number in base 2. Or if you prefer, the bits should be set as follows. Call the bits (or LED's) from left to right bit1, bit2, bit3, bit4 and the number n. Then

- Bit1 is on iff (n)& 1
- Bit2 is on iff (n>>1) & 1
- Bit3 is on iff (n>>2) & 1
- Bit4 is on iff (n >> 3) & 1

Remember that the first argument to a shell script is in the variable $1 and that the let command supports both shift and bitwise and operators.

Test setbits.sh by running the following examples. What should they look like when you run them

- ./setbits.sh 0
- ./setbits.sh 1
- ./setbits.sh 2
- ./setbits.sh 4
- ./setbits.sh 8

countTo15.sh - count from 0 to 15 in a loop, calling setbits.sh with each argument and then waiting for 1 second after each call. If you display 16, what do you get in the bits?

cylon.sh - implement a Cylon warrior eye using your four bit display. (See here.) One LED should be powered, and it should move from left to right and back again. Each position should be held on for 0.25 seconds. Note: the sleep command in Raspian works for fractional seconds. Be careful at the ends of the sequence not to wait too late. As this command never ends you will have to kill it using ^C

blinkenlights.sh - Historically, computer hardware was typically equipped with blinking status lights. This resulted in the concept of 'blinkenlights' (see here). And warning signs such as that shown below

: **ACHTUNG!**

ALLES TURISTEN UND NONTEKNISCHEN LOOKENPEEPERS!
DAS KOMPUTERMASCHINE IST NICHT FÜR DER GEFINGERPOKEN UND MITTENGRABEN! ODERWISE IST EASY TO SCHNAPPEN DER SPRINGENWERK, BLOWENFUSEN UND POPPENCORKEN MIT SPITZENSPARKEN.
IST NICHT FÜR GEWERKEN BEI DUMMKOPFEN. DER RUBBERNECKEN SIGHTSEEREN KEEPEN DAS COTTONPICKEN HÄNDER IN DAS POCKETS MUSS.
ZO RELAXEN UND WATSCHEN DER BLINKENLICHTEN.

Write code to randomly assign a bit pattern to your array of LED's changing every 0.25 of a second. Note: the Bash variable $RANDOM (note: upper case) is a random number.

Grading

To obtain a grade for this lab you must show your TA your Raspberry Pi running the code that you developed and that your code has been properly added your own GitHub repository. Your TA will ask you to demonstrate the 10 commands you have added.