This assignment does not count toward the final grade.

Final Exam - Practice Code

Submit Assignment

Due No Due Date **Points** 0 **Submitting** a file upload **File Types** gz and tgz

Practice Code for the Practical Portion of the Final Exam

The practical portion of the exam will use portions of this "practice" source code, so I urge you to download and experiment with this source code before the final exam.

In this practice code I've supplied some of the code that you will be required to write during the exam. For example, I supplied a Makefile with this practice code, but during the final exam you will be required to create a Makefile yourself.

For the practical you should know how to use the `rpi3b_accessory` class, the `wiringPi` library, and the C++ classes specified herein to perform the these tasks:

- Create and register interrupt service routines (ISR) that handle switch actuation events on the accessory board--i.e., pushbutton "press" events (logic LOW->HIGH transitions), and slider switch actuation events (logic LOW->HIGH events **and** HIGH->LOW events).
    - **:: IMPORTANT ::** I created/added a method to the rpi3b_accessory class named `RegisterISR`. This method's parameters are the same as for function wiringPiISR() within the wiringPi library. Use this **rpi3b_accessory::RegisterISR** method to register your interrupt service routine functions in function main. This is a much simpler approach for registering your ISR functions with the wiringPi library and will save time during the practical exam. See the example code in function **main()** within file **main.cc**.
- Create and use instances of the C++ standard library class `std::atomic<bool>` (a.k.a., `std::atomic_bool`) as the means of signaling switch actuation events from the interrupt service routines to other threads of execution within your program.
- Create and use instances the C++ standard library classes `std::mutex` and `std::lock_guard<T>` to properly implement critical sections that serialize access to a shared resource (which will be `std::cout`) across multiple threads of execution.
- Pushbutton and Slider Switches
    - EXAMPLE
        - In function main(), terminate execution of a do-while loop when the user presses pushbutton #5.
- Light-Emitting Diodes
    - Turn ON|OFF individual light-emitting diodes (LED) on the accessory board.
    - Read the ON|OFF state of an LED on the accessory board. See method `rpi3b_accessory::ledRead()` in file **rpi3b_accessory.h**.
    - EXAMPLES

- Each press of pushbutton #2 toggles the ON|OFF state of the yellow LED.
- Ensure the green LED's ON|OFF state matches the ON|OFF state of slider switch #2.

- The 7-Segment Display
  - Turn OFF|ON the 7-segment display. See methods `rpi3b_accessory::displayOff()` and `rpi3b_accessory::displayOn()` in file **rpi3b_accessory.h**.
  - Display decimal digits 0 through 9 on the 7-segment display. See method `rpi3b_accessory::displayWrite()` in file **rpi3b_accessory.h**.
  - EXAMPLES
    - Each press of pushbutton 4 increments the number displayed on the 7-segment display.
    - Each press of pushbutton 3 decrements the number displayed on the 7-segment display.

Quick Start

PRACTICE these instructions BEFORE THE FINAL so that you understand this workflow.

Use this "dummy" assignment to practice downloading your tarball file from Canvas before the day of the final exam. You can download your tarball multiple times via this dummy assignment.

Perform these instructions on the **desktop computer,** NOT on the Raspberry Pi.

1. Open a command shell window.
2. Use your computer's mouse to **select ALL lines in Listing 1 below**. Do not edit or modify the commands shown in Listing 1.
3. Position the mouse's cursor over the command shell window and then click on the mouse's scroll wheel; this pastes the commands from Listing 1 onto the command shell window.
4. Press the keyboard's ENTER key once.
5. The practice code is provided within **the desktop computer's file system** in folder `${HOME}/rpifs/home/pi/ece3220-practice-$USER/` where $USER is your Mizzou pawprint. *(Note that within the Raspberry Pi's file system this folder is located at `/home/pi/ece3220-practice-$USER/`.)*

Listing 1. Quick start commands to be invoked on the desktop computer, not on the Raspberry Pi.

```
mount.rpifs -q
cd ~/rpifs/home/pi/
wget -qO- http://fabrica.missouri.edu/ece3220-practice.tar.gz \
  | tar xz --transform "s@^ece3220-practice@ece3220-practice-${USER}@"
cd ece3220-practice-$USER
# select this line too
```

Deliverable

PRACTICE these instructions BEFORE THE FINAL so that you understand this workflow.

Use this "dummy" assignment to practice uploading your tarball file to Canvas before the day of the final exam. You can upload your tarball multiple times to this dummy assignment.

Perform these instructions on the **desktop computer,** NOT on the Raspberry Pi.

1. Open a command shell window.
2. Use your computer's mouse to **select ALL lines in Listing 2 below**. Do not edit or modify the commands shown in Listing 2.
3. Position the mouse's cursor over the command shell window and then click on the mouse's scroll wheel; this pastes the commands from Listing 2 onto the command shell window.
4. Press the keyboard's ENTER key once.
5. On **the desktop computer**, in your `HOME` directory, you should now have a gzipped tarball file named `ece3220-practice-$USER.tar.gz` where $USER is your Mizzou pawprint.
6. Use the **SUBMIT ASSIGNMENT** link provided with this assignment to upload to Canvas the gzipped tarball file `ece3220-practice-$USER.tar.gz`.

Listing 2.   Deliverable commands to be invoked on the desktop computer, not on the Raspberry Pi.

```
mount.rpifs -q
cd ~/rpifs/home/pi/ece3220-practice-${USER}/
make dist
cd
ls ece3220-practice-${USER}.tar.gz
tar tf ece3220-practice-${USER}.tar.gz
# select this line too
```

Reference

wiringPi Documentation

- **wiringPi Setup Functions (http://wiringpi.com/reference/setup/)**
  - wiringPiSetup
- **Core Functions (http://wiringpi.com/reference/core-functions/)**
  - pinMode, digitalRead, digitalWrite
- **Priority, Interrupts and Threads Functions (http://wiringpi.com/reference/priority-interrupts-and-threads/)**
  - wiringPiISR

Accessory Board Schematic Diagram

You might need **the accessory board's schematic diagram** (Fig. 1).

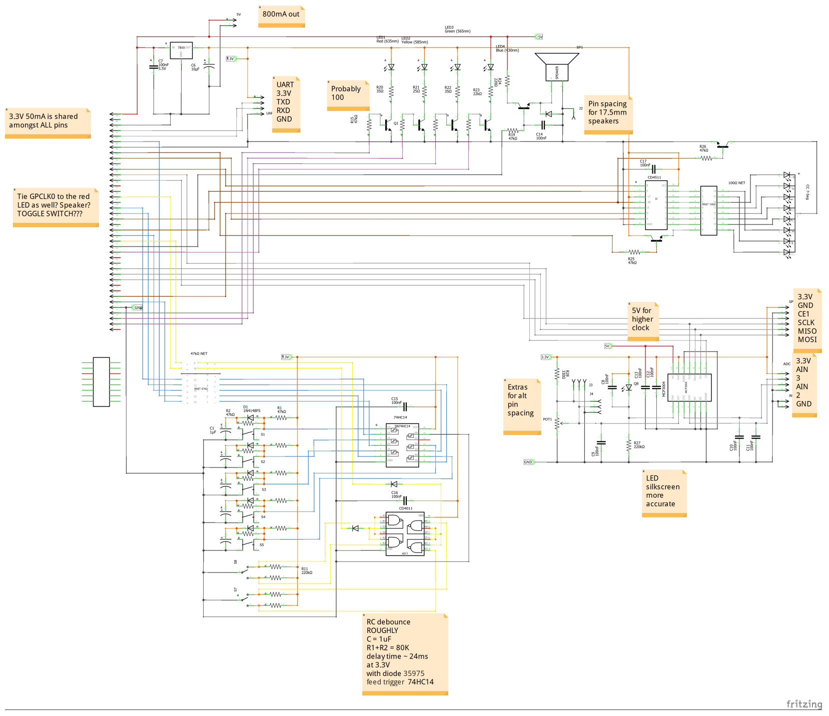
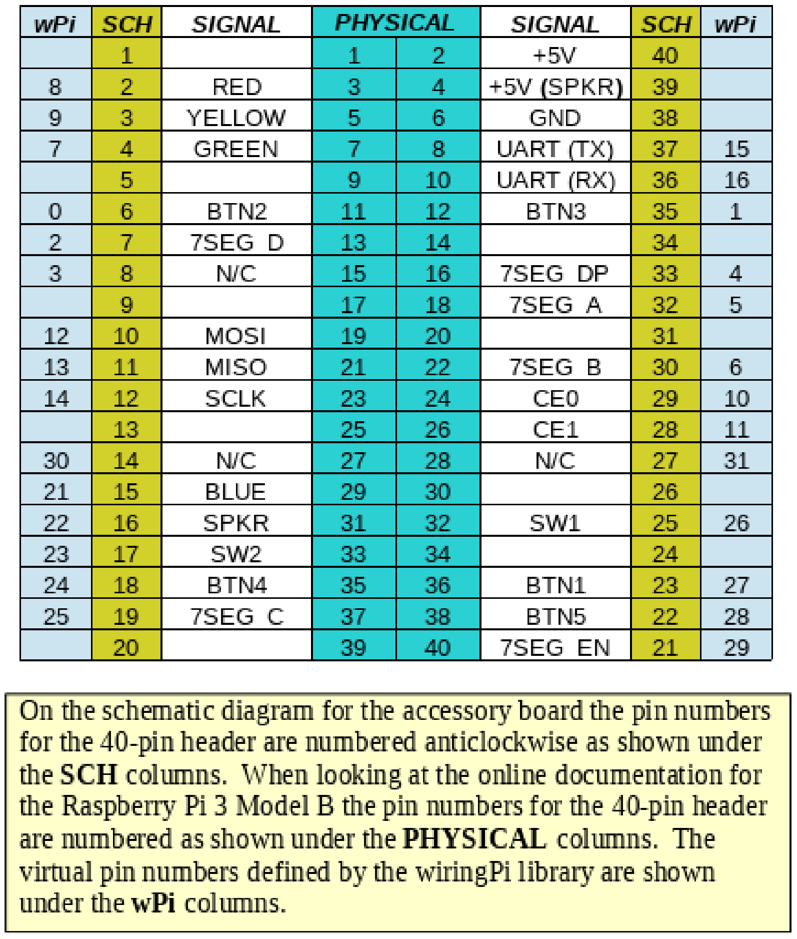


Figure 1. Schematic diagram for the Raspberry Pi 3B accessory board.

Accessory Board Pin Mappings

In Figure 2 the pin numbers shown under the columns labeled **SCH** correspond to the anticlockwise pin numbering scheme for the Raspberry Pi's 40-pin header as shown on the accessory board's schematic diagram (Figure 1). The pin numbers shown under the columns labeled **PHYSICAL** correspond to the odd-even pin numbering scheme for **the 40-pin "GPIO EXPANSION" header J8 as shown on the Raspberry Pi 3 Model B+ "Reduced Schematic" diagram (https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/rpi_SCH_3bplus_1p0_reduced.pdf)**. The pin numbers shown under the columns labeled **wPi** correspond to the virtual pin numbers the wiringPi software library assigns to the Raspberry Pi's physical pin numbers as shown under the **PHYSICAL** column.



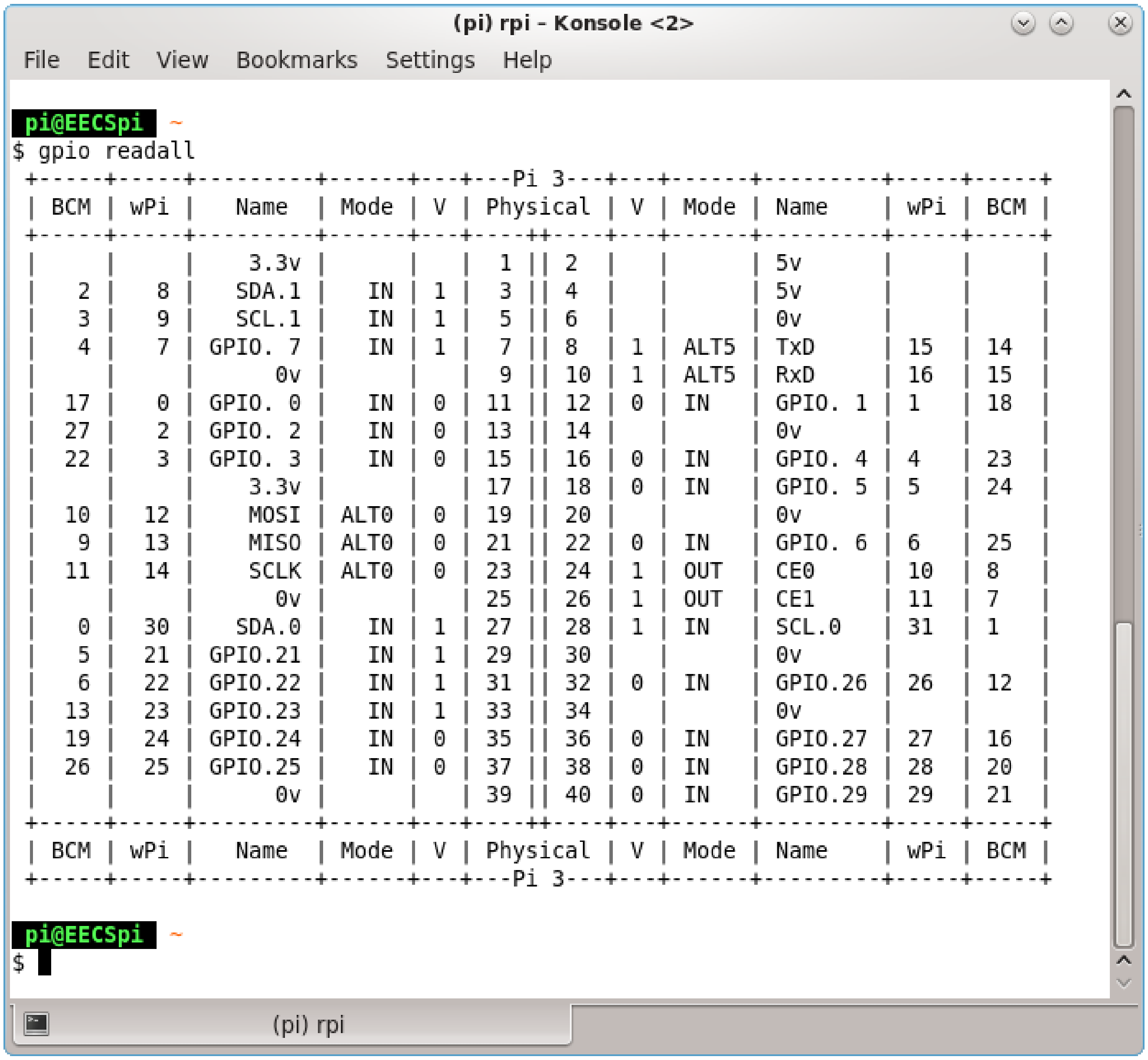
wPi	SCH	SIGNAL	PHYSICAL		SIGNAL	SCH	wPi
	1		1	2	+5V	40	
8	2	RED	3	4	+5V (SPKR)	39	
9	3	YELLOW	5	6	GND	38	
7	4	GREEN	7	8	UART (TX)	37	15
	5		9	10	UART (RX)	36	16
0	6	BTN2	11	12	BTN3	35	1
2	7	7SEG D	13	14		34	
3	8	N/C	15	16	7SEG DP	33	4
	9		17	18	7SEG A	32	5
12	10	MOSI	19	20		31	
13	11	MISO	21	22	7SEG B	30	6
14	12	SCLK	23	24	CE0	29	10
	13		25	26	CE1	28	11
30	14	N/C	27	28	N/C	27	31
21	15	BLUE	29	30		26	
22	16	SPKR	31	32	SW1	25	26
23	17	SW2	33	34		24	
24	18	BTN4	35	36	BTN1	23	27
25	19	7SEG C	37	38	BTN5	22	28
	20		39	40	7SEG EN	21	29

On the schematic diagram for the accessory board the pin numbers for the 40-pin header are numbered anticlockwise as shown under the **SCH** columns. When looking at the online documentation for the Raspberry Pi 3 Model B the pin numbers for the 40-pin header are numbered as shown under the **PHYSICAL** columns. The virtual pin numbers defined by the wiringPi library are shown under the **wPi** columns.

Figure 2. The pin numbers for the Raspberry Pi's 40-pin header.

[rpi]$ gpio readall

When you are logged on to the Raspberry Pi you can use the command `gpio readall` to display an "ASCII art" table that shows the mappings between the wiringPi virtual pin numbers (**wPi**), and the physical pin numbers on the Raspberry Pi's 40-pin header (**Physical**), and the pin numbers on the Broadcom BCM2837 SoC microprocessor IC (**BCM**); see Figure 3.



```
pi@EECSpi ~
$ gpio readall
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5v      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT5 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT5 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI | ALT0 | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO | ALT0 | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK | ALT0 | 0 | 23 || 24 | 1 | OUT  | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | OUT  | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 1 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+

pi@EECSpi ~
$ 
```

Figure 3. An example of the ASCII art table that's produced when you run the command `gpio readall` on the Raspberry Pi.