

## Lab 5: Serial – Parallel Conversion

Or: How shifty is that register?

Due: **Beginning of class**, Tue Feb 29

### What you will do:

1. Determine key operating parameters for a typical shift register.
2. Achieve basic control of a shift register
3. Implement a bar graph level meter, consisting of 14 LEDs.
4. Synchronize the bar graph display with a 2-digit numerical display
5. Do some cool stuff.

### What you need to submit and when:

1. There is no pre-lab for this week's lab.
2. Complete the demo of:
  - (1) bar graph level meter, displaying a pattern smoothly varying between 0 → full → 0
  - (2) a synchronized bar graph and digital display, driven programmatically (Task 4)
  - (3) a synchronized bar graph and digital display, to monitor a physical quantity (Task 5)
3. The post-lab **within a week** of the due date of this lab (ie. **before** the beginning of that class).

### Marks:

Each of the demos identified above are weighted equally, and combined to generate your mark for the in-lab portion of this week's lab.

### Required Equipment:

- Computer with Arduino IDE & Teensy extensions installed and working
- Teensy board and USB cable
- All components required for the previous lab (7-segment displays)
- 8-bit shift register (2 x SN74HC595)
- LEDs: green (8 x), yellow (4 x), red (2 x)
- Resistors – suitable for maintaining LED currents to the limits specified for the shift register

### References and Resources:

- You will need to combine this lab with the circuitry and code from the previous lab. Try to save as much of your work as is reasonable!
- Reminder of the "resistor colour code" (BBROYGBVGW, silver, gold)  
English: Bad Beer Rots Our Young Guts, But Vodka Goes Well  
French: Ne Mangez Rien Ou Jeuner, Voilà Bien Votre Grande Betise  
(Noir, Marron, Rouge, Orange, Jaune, Vert, Bleu, Violet, Gris, Blanc)
- Course textbooks: Chapter 6, "Beginning Arduino"
- PartsList-Explanations file with part numbers, for referencing data sheets
- Data sheet for the shift registers
- familiarity with the analogRead() function in Arduino

## Task 1: Determine Operating Parameters for Shift Register

It's important to know what are the required voltage & current parameters for the shift register IC. Find the data sheet for the devices in your kit.

- Step 1. Using the supplied parts list, identify the manufacturer and Digikey part number.
- Step 2. Go to Digikey's website: [www.digikey.ca](http://www.digikey.ca) and search on the part number.
- Step 3. Navigate through the results to find the link to the datasheet and download it.
- Step 4. Identify the pinout and the marking which designates pin 1.
- Step 5. Identify the appropriate voltage & current parameters.
- Step 6. Use Ohm's law and other formulae as required to determine a suitable value for the current limiting resistors which will be used to drive the green, yellow, and red LEDs.

## Task 2: Verify Correct Operation

You will need to combine this lab with the circuitry and code from the previous lab. Try to save as much of your work as is reasonable. The steps below serve to verify basic operation of the shift register. If you make your code modular, and document both the code and wiring, you will be able to re-use most of it for later tasks in this lab.

- Step 1. Using the data sheet from the previous task, plan out the placement of a pair of shift registers with a set of 14 LEDs and their accompanying resistors. Note that you will need to save space for the dual 7-segment displays from last lab, since they will be needed in later steps.
- Step 2. Identify the shift register signals that you will need to control via the Arduino. As a minimum, you should consider the SERIAL input (SER), the Shift Register clock (SRCLK), and the output Register clock (RCLK).
- Step 3. Wire up a single shift register and a set of 8 green LEDs.
- Step 4. Start simple: choose a random pattern of On/Off for 8 LEDs, and verify your ability to control a single shift register by creating that pattern. Consider carefully whether the LSB or MSB needs to be loaded first!
- Step 5. Implement the code to light up a single LED and make it "travel", bouncing from end to end at some pleasing speed.

## Task 3: Full Bar Graph Display (Demo)

The objective of this task is to implement a bar graph display which spans 14 LEDs and two shift registers. Similar to the previous lab where you needed an independent "master control" for each 7-segment display, you'll need a master control signal to load each shift register independently.

- Step 1. Wire up the second shift register with an additional 4 yellow LEDs and 2 red LEDs. The colours should be arranged to form a continuous pattern from left → right going from green → yellow → red. What control signal will you use as the master control signal to differentiate the first & second shift register?
- Step 2. Start simple: choose a random pattern of On/Off for the LEDs, and verify your ability to control the pair of shift registers by creating that pattern. Consider carefully whether the LSB or MSB needs to be loaded first.
- Step 3. Implement a bar graph which displays a pattern smoothly varying between 0 → full → 0. The display should change at a speed which is clearly discernible.

## Task 4: Synchronized Bar Graph and Digital Display (Demo)

This task extends the previous one by synchronizing a digital readout (2 x 7 segment display) with the bar graph.

- Step 1. Starting from the last step in the previous task, modify your circuit and code as necessary so that the bar graph and digital display are in sync. You will likely need to adjust the speed of the pattern so that digital value is easily discernible.
- Step 2. Sit back and pat yourself on the back for a job well done. Get ready to tackle the next task.

## Task 5: Extend: Choose one option from below. (Demo)

Even a few extra minutes is enough to add some neat extra functionality. Please ask for further details if you are unsure what any of these options requires!

- Step 1. Add a potentiometer and use the `analogRead()` function to determine its value. Display the position of the thumbwheel.
- Step 2. Make a simple game by adding a switch. The player has to press the button exactly when the value reaches the boundary between the green and yellow LEDs (ie. 57%). You need to crank up the speed from the previous task to make it more challenging.
- Step 3. Make a simple thermometer by adding a thermistor (item 26 in your kit); use the `analogRead()` function to determine its value. Display the temperature expressed as the number of degrees (with one decimal in the digital display) above a “minimum comfortable temperature” of 16 C. You’ll need to determine appropriate thermistor values by comparing to a real thermometer.
- Step 4. Make a simple light meter by adding a light sensor (item 17 in your kit); use the `analogRead()` function to determine its value. After testing, you’ll be able to determine reasonable values for (mostly) dark and (full) bright; use those for your “0” and “full” values.
- Step 5. Make a simple battery tester, with wires for connecting the battery. Use the `analogRead()` function to determine its value. Display the voltage as a value (with two decimals in the digital display) above a “minimum acceptable voltage” of 1.0V. You’ll want to get some extra info from the Professor about a suitable choice of a load resistor.
- Step 6. Can you find something else that produces a small voltage (eg. 0-5V) or a variation in resistance that you can hook up to the Arduino? Display the value of the voltage or resistance according to a suitable scale. For example, a microphone (or a speaker used as a microphone!)