ROB-GY 6103 Advanced Mechatronics: HW 2

Problem 1: Connect one terminal of a push button to P8 of an Arduino UNO through a 220Ω resistor. Connect the other terminal of the push button to the Arduino GND. Connect an LED to P9 with an appropriate resistor. Write an Arduino UNO program where the void setup() function includes commands that configure P8 as an input pin (in appropriate configuration) for digital sensing and P9 as an output pin. Next, the void loop() function must include commands that turn the LED ON when the button is pressed and OFF when the button is released. First do all work using high-level Arduino commands and then redo this using ATMEL register/port manipulation (2 instructions in the setup function and a single instruction in the loop() function).

Problem 2: Connect a Seven-Segment common cathode display to pins P4 to P11 of an Arduino UNO (P4: Segment B, P5: Segment DP, P6: Segment A, P7: Segment C, P8: Segment F, P9: Segment D, P10: Segment G, P11L Segment E). Write an Arduino UNO program that:

- Displays numerals 0—9 on the Seven-Segment display in sequence with a one second pause between each numeral. In doing this, use appropriate bitmasks to store the segment ON/OFF pattern for each numeral.
- After displaying the sequence 0—9, turn off the display for two seconds and then display the sequence in reverse order, i.e., 9—0.
- Make sure that your program efficiently uses the bitwise operations to control the various segments of the display.

Problem 3: Write an Arduino UNO program that does the following.

1. Uses the USART serial communication to prompt a user to enter a number between 0—4096 on the serial console and receives the same.
2. Next, the program uses bit shift operation to scale the user entry to a range between 0—256. Moreover, this scaled value is saturated to 255 (i.e., maximum value 255).
3. The saturated-scaled value is used to drive a PWM-capable pin of Arduino UNO to control the brightness of an LED connected to it.
4. Serial console displays in a well-formatted manner, the raw user input, its scaled representation, and the value used for the PWM output.
5. Your program repeats the process every 5 seconds (user input, scaling, saturation, PWM, etc.).

Problem 4: Connect the 5V and GND wires of a Parallax PING sensor to the 5V and GND pins of an Arduino UNO. Connect the signal wire to P9 of Arduino. Next, connect an LED to P10 of Arduino using an appropriate resistor. Write a program that performs as outlined below.

1. Measures the distance to an object using the PING))) sensor in cm.
2. Controls the LED based on the distance:
   - ON if distance < 10 cm.
   - Blinking at 1 Hz if distance is between 10 cm and 50 cm.
   - OFF if distance > 50 cm
3. Prints the measured distance in centimeters on the serial terminal every 500ms.
4. Uses non-blocking timing (“millis()”) for LED blinking and distance updates.

Problem 5: Connect 5V, GND, and RX pins of a Parallax LCD to the 5V, GND, and P7 of an Arduino UNO. Next, connect three buttons (BTN0, BTN1, BTN2) to P8, P9, and P10 of Arduino in an appropriate manner. Write a program that allows a user to input four digits (D_1, D_2, D_3, D_4) that are used to represent

a floating-point number ($D_1D_2.D_3D_4$) between 00.00 and 100.00 (up to two decimal places) using the three buttons as follows.

- BTN0: Shift between digits
- BTN1: Increment the current digit (cycling from 0 to 9)
- BTN2: Finalize the input

Once the user finalizes their input, the program should compute the square root of the number, round it to two decimal places, and display it on the LCD. After 5 seconds, the display should return to input mode. Ensure the baud rate of the LCD is correctly configured for communication with the Arduino. Also comment on the communication method that the LCD uses.

Problem 6: Connect an LED to P9 of an Arduino UNO using an appropriate resistor. Write an Arduino UNO program that allows a user to control the LED via the serial monitor by sending the following commands.

- "ON" → Turns the LED on
- "OFF" → Turns the LED off
- "BLINK" → Makes the LED blink at a 1-second interval
- "STOP" → Stops blinking and turns the LED off

The Arduino should acknowledge the command by printing an appropriate response in the terminal.