George Fox University
Engineering Principles II
ENGR 152

Detailed Program Design – Pseudo Code

Libraries and Variables

- **Wire.h** allows for communication to I2C devices, like our display
- **Button.h** custom library made by Jack Sides to read and debounce a given button
    - https://drive.google.com/file/d/1qP-ozoc1w1TRmiRVgIl0BS0s_7BR7eC9/view?usp=share_link
- **hd44780.h** main library for any LCD hd44780 display
    - https://github.com/duinoWitchery/hd44780
- **hd44780ioClass/hd44780_I2Cexp.h** library included in the main hd44780 that controls LCD displays with I2C expansion pack
- **Stepper.h** built in library for stepper motor controls
- *CLK* defined int for output B of rotary encoder
- *DT* defined int for output A of rotary encoder
- *SW* defined int for button pin of rotary encoder
- *PIEZO* int for the piezo pin
- *toneLength* int for the length of a given tone in ms
- *stepsPerRevolution* constant integer for the number of steps per revolution of the stepper motor
- *oneN1* constant integer that stores the pin for the 1st magnet field of stepper
- *oneN2* constant integer that stores the pin for the 2nd magnet field of stepper
- *oneN3* constant integer that stores the pin for the 3rd magnet field of stepper
- *oneN4* constant integer that stores the pin for the 4th magnet field of stepper
- *currentStateSW* integer to store the state of the rotary button
- *lastStateSW* int to store the last state of the rotary button for next loop
- *currentStateCLK* int to store the current state of output B
- *lastStateCLK* int to store the last state of output B
- *step* int to keep track of the position of the cursor
- *state* int to keep track of the main menu state machine
- *substate* int to keep track of the substate state machine
- *devMode* bool to check whether the developer mode is enabled/disabled
- *versesLength* constant int for length of the verse

- *verseOfDay* string holding the given verse of the day
- *minutes* int to hold current clock minutes
- *hours* int to hold current clock hours
- *days* int to hold days passed
- *previousDays* int to hold the # of the previous day
- *zeroStepperState* int to move through the zero stepper function state machine

Arrays:

- *uint8_t curs1[8]* Hex Array storing custom character to display to lcd
- *uint8_t curs2[8]* Hex Array storing custom character to display to lcd
- *Uint8_t curs3[8]* Hex Array storing custom character to display to lcd
- *uint8_t anim1[8]* Hex Array storing custom character to display to lcd
- *uint8_t anim2[8]* Hex Array storing custom character to display to lcd
- *uint8_t anim3[8]* Hex Array storing custom character to display to lcd
- *uint8_t chev[8]* Hex Array storing custom character to display to lcd
- *String verses[]* Array of strings storing bible verses
- *String mainMenuItems[]* Array of strings holding main menu

---

Setup

1. Initialize stepper blindStepper
    a. Input stepsPerRevolution for step count
    b. Input oneN1, oneN2, oneN3, and oneN4 for the digital output pins allocated to the stepper
    c. Set the stepper speed to 15
2. Initialize Buttons
    a. Use backButton pin as input for back button
    b. Use SW pin for the rotarySwitch button
3. Begin Serial Port
    a. Use Baud rate 9600
4. initialize the LCD
    a. If failed print to console that it failed
5. Create characters
    a. For curs1,2,3
    b. For anim1,2,3
    c. For chevron cursor
6. Initialize pins
    a. CLK pin as an input
    b. DT pin as an input

    c. SW pin as an input pullup
7. Read the last state of the CLK pin using digitalRead
8. Set verse of the day
    a. Set verseOfDay string equal to a random selection from the verses array
    b. Set that verse to the first position of the mainMenuItems array
    c. Display mainMenuItems to lcd

Loop

Peripherals:

- Read light Level from Light sensor
    - Call analogRead() to read the sensor
- Update Time
    - Read from Millis() gets amount of milliseconds passed since boot of device
    - Set Seconds, Minutes, Hours, and Days based on the milliseconds passed
- Read Button
    - Check if boolean BackButton.IsPressed() is true from Button Library
- Read Rotary Encoder
    - Call checkRotary() function

State Machine 1: Display Items

- **Print Main Menu:**
    - Call printMainMenu()
    - Will only leave here if the user presses to enter one of the subMenus
- **Print Modes Menu:**
    - Call printModesMenu()
    - Will only leave if the user goes back to main menu or goes further down menu tree
- **Print Time Set Menu:**

    - Call printTimeSetMenu()
    - Will only leave if the user goes back to main menu or goes further down menu tree
- **Print Light Menu:**
    - Call printLightMenu()
    - Will only leave if the user goes back to main menu or goes further down menu tree
- **State Name:** Each name should be descriptive of what will occur in that state
    - **o** Each bullet should have sub-notes of what the state will do each time it is run in the main loop
    - **o** Include notes that explain what causes the state machine to leave the state, along with what will be done as the state changes
    - **o** References to functions can be noted as: call function()

State Machine 2: Print Modes Submenu

- **Print to LCD:**
    - Print on each line, “Settings”, “Zero Stepper”, and “Open/Close”
    - Will only exit if the user presses back button or moves into another submenu of this.
- **Zero Stepper:**
    - Call State Machine 3: Zero Stepper
    - Will only exit if the user completes the calibration or presses back
- **Open/Close:**
    - Print to screen on each line “open blinds” , “closed blinds” , “swipe”

State Machine 3: Zero Stepper

- **LowerBound:**

    - Prompts the user to press the manual override button to turn the blinds device until it is fully closed.
    - Sets the stepperPosition integer to 0
    - The next state will only run if the user presses the rotary encoder or the back button
- **Upper Bound:**
    - Prompts the user to move the blinds to open position and press in
    - Reads the position of the stepper motor, and controls it with the rotary encoder
    - If the rotary switch button is pressed, record the value of the stepper in the upperBound variable
    - The next state will only run if the user presses the rotary encoder or the back button

---

Custom Functions

- scrollMenu()
    - **Inputs**: none
    - Check if rotary encoder has turned clockwise or counterclockwise
        - If clockwise, scroll down and increase the step variable
        - If counterclockwise, scroll up and decrease the step variable
    - **Output:** none
- checkState()
    - **Inputs**: none
    - Call scrollMenu()
    - Checks if the user is in a main menu or submenu
        - If they press a button call selectLine(bool subState)
    - If no button is pressed simply print the given function currently at through calling findFunction()
    - **Output:** none
- displayItems()
    - **Inputs**: none

    - Clears the lcd
    - Calls State Machine 1: Display Items
    - **Output:** none
- updateTime()
    - **Inputs**: none
    - Calls millis() to determine the exact time in minutes, hours, and days
    - If the day has changed, set the verse of the day to a new verse
    - **Output:** none
- printMainMenu()
    - **Inputs**: none
    - Clear lcd
    - Print on each new line each string from mainMenuItems[] array
    - **Output:** none
- printTimeSetMenu()
    - **Inputs**: none
    - Set cursor to (0,0)
    - Print “Time set Menu”
    - **Output:** none
- printModesMenu()
    - **Inputs**: none
    - Call State Machine: Print Modes Submenu
    - **Output:** none
- printLightMenu()
    - **Inputs**: none
    - Prints to screen “Light level menu”
    - Will act as the submenu for the light menu
    - **Output:** none
- selectLine()
    - **Inputs**: boolean isInSub

    - Write the custom cursor symbol on the line that the user is currently hovering over
    - Update the step for both the submenu and main menu to match the users position
    - Call displayItems()
    - Play tone through piezo for moving lines
    - **Output:** none
- returnButtonPressed()
    - **Inputs**: isSubMenu
    - Determines by where the user is at in the menu tree
    - If the user presses the back button
        - Move back one single menu level
    - **Output:** none
- findFunction()
    - **Inputs**: none
    - Call switch statement that will run through all possible position options for the user's position and call the appropriate function to do an action
    - Not all are created yet, but currently zeroStepperCall(), controlBlinds() are created for the settings submenu
    - **Output:** none
- zeroStepperCall()
    - **Inputs**: none
    - Call State Machine 3: Zero Stepper
    - **Output:** none
- controlBlinds()
    - **Inputs**: none
    - Checks what menu / submenu the user is in
    - If the user selected the Open/Close function in settings and they pick one of the three options

    - Either move stepper open with rotateStepper(upperBound), or close the blinds by zeroing the stepper, or swiping between the two if the last option is selected.
    - **Output:** none
- checkRotary()
    - **Inputs**: none
    - Reads the rotary encoder
    - If the encoder is turned clockwise, return 1
    - If counterclockwise, return -1
    - Otherwise return 0
    - **Output:** Integer return Value
- rotateStepper()
    - **Inputs**: targetPosition
    - If the stepperPosition is not the Target Position
        - Move the difference of the values to get to the target position
        - Then set the current position to the target position
        - Then write all the digital pins going out to the stepper motor to low to save energy.
    - **Output:** none