ECEN 5823 Spring 2025

Assignment 9: BLE Client with Security

Objective: Modify your Client implementation to use Numeric Pairing for additional security and MITM protection. Verify your implementation can successfully display the state of the Server's encrypted characteristic on the Client LCD once the devices are bonded.

Build a command table to show the commands you used to implement the A7 Client plus the commands you will use for this assignment.

Note: This assignment will start with the completed code from Assignment 8, configured as a Client.

Instructions:

1. Start by creating your assignment repository using the assignment link at https://classroom.github.com/a/be9oLGLE. You will not need to clone this from the GitHub repo you just created into a new local directory since you will be starting with your code from the previous assignment. Instead, follow these instructions and run these commands with git bash within your assignment directory. Ensure you are setup to use SSH keys before proceeding, see the instructions from github at https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/ and be sure to use Git Bash For Windows or Cygwin if using a Windows environment. (Cygwin .ssh directory will be different than Git Bash.)

    a. On your local PC, duplicate the folder created in Assignment #8 to a new folder, name it something like ecen5823-assignment9-<username> where <username> is your GitHub username.
    b. Change your current directory to this new folder ecen5823-assignment9-<username> and execute the following git commands.
    c. git remote remove origin
    d. git remote add origin <url>
        i. Where <url> is the URL for the repository created with the link above
        ii. This adds the new submission repository created in the link above as your new origin.
    e. git push origin master
    f. Import ecen5823-assignment9-<username> into Simplicity Studio
        i. Remember to perform a **Clean…** immediately after you import the project into Simplicity Studio.

2. In the questions folder, copy the file “Assignment 6 - Client Command Table for A7.xlsx” into a new file named “Client Command Table A9.xlsx”.
    a. Extra credit (+10 points) is available if you include a bubble state diagram of your discovery state machine in your command table spreadsheet for A9.

3. Design your program: Using your A7 and A8 command tables as a guide, create your A9 command table to include all of the events you will need to handle and commands you will need to add in order to implement the bluetooth “Numeric Comparison” pairing process for a Client. All existing A7 functionality shall be maintained, so you can start with your A7 command table and add to it.

4. Implement the Client commands as described in your A9 Command Table.
    a. The “Responder” will be the Server, your A8 Server code or A9 code configured as a Server.
    b. The “Initiator” will be the Client.
    c. The Client shall use the **DisplayYesNo** IO capabilities option to display the passcode on the LCD for verification, and use PB0 as the confirmation (Yes button).
    d. On each boot or open event, your Client’s discovery state machine shall run through a discovery sequence and enable HTM and button_state indications. Your Client shall leave HTM indications enabled. The state of indications enabled shall be reflected with LED0 and LED1. LED0 shall be On when HTM indications are enable, and off when disabled. LED1 shall be On when button_state indications are enabled, and off when disabled. These requirements apply to your A9 Server build, see images below.
    e. Your Client shall use Push Button PB1 to issue a GATT characteristic read of the encrypted button_state characteristic on the Server. This is the same functionality as the the “Read button” on the mobile phone app and does not depend on whether the device is bonded or not. Every press of PB1 by itself shall send a GATT characteristic read of the encrypted button_state characteristic to the server. If a read of the encrypted button_state characteristic is in-flight, and you get a 2nd press of PB1, you can skip/ignore/ not-send the read command (there is no requirement to enqueue these GATT read commands in a circular buffer).
    f. Your Client shall use a sequence of PB1 and PB0 presses and releases (press PB0, then press and release PB1, then release PB0) to toggle the the state of the button_state characteristic indication enable on your Server. Each time you press this sequence of PB0 and PB1, the button_state indication enable on your Server should toggle, and be reflected by LED1 on your Server. Also note this sequence of pressing PB0 and PB1 shall not send a GATT characteristic read of the encrypted button_state characteristic on the server. This PB0 and PB1 pressing sequence mimics toggling the indication enable (upright gray/blue arrow) in the EFR Connect App.
    g. The Silicon Labs documentation on Numeric Comparison at: https://docs.silabs.com/bluetooth/latest/general/security/pairing-processes#numeric-comparison isn’t complete. See below for an updated flowchart.

h. Your Client shall discover your Server at the SERVER_BT_ADDRESS, just like in A7, connect to it and display the Server's bd_addr on row DISPLAY_ROW_BTADDR2 of the LCD. When discovering, DISPLAY_ROW_CONNECTION shall display "Discovering". When connected, DISPLAY_ROW_CONNECTION shall display "Connected".
i. Once connected:
    a. Pressing PB1 on the Client shall send 1 sl_bt_gatt_read_characteristic_value() command to read the encrypted button_state characteristic on the Server. The first occurrence of this GATT read command will fail with error code = 0x110F (SL_STATUS_BT_ATT_INSUFFICIENT_ENCRYPTION, the attribute requires encryption before it can be read). Your Client shall then respond to this error response and raise the security level by calling sl_bt_sm_increase_security(). This will trigger the Server and Client into the bonding process
j. Your Client shall then display "Passkey XXXXXX" on the DISPLAY_ROW_PASSKEY line and "Confirm with PB0" on the DISPLAY_ROW_ACTION line. Same thing should occur on your Server. Press PB0 on your Server to complete bonding.
k. Press PB0 on the Client to complete bonding on the Client. The Client should complete the pairing process and clear the DISPLAY_ROW_PASSKEY and DISPLAY_ROW_ACTION lines. Use interrupts for PB0 and PB1.
l. Once bonding is established, calls to sl_bt_gatt_read_characteristic_value() will return event = sl_bt_evt_gatt_characteristic_value_id with an att_opcode of sl_bt_gatt_read_response. The value returned shall be used to display the "Button Pressed" and "Button Released" messages on your Client as shown below.
m. Use sl_bt_sm_delete_bondings() on boot or connection closed to prove you can complete the bonding procedure successfully. This ensures we don't save bonding information in the Gecko, and forces a re-pair each time we connect. You wouldn't do this on a real-world production design, but this allows us to easily test reconnection scenarios.
n. With button_state indications disabled on your Server, when PB0 is pressed and held on the Server and then pressing PB1 on the Client, your Client should display "Button Pressed" on row LCD Row 9. When PB0 is released on the Server and then pressing PB1 on the Client, your Client should display "Button Released" on row LCD Row 9.
o. Add displayPrintf() to display A9 on row DISPLAY_ROW_ASSIGNMENT
p. Your Server implementation should still be functional and match Assignment 8 behavior when your project is built with DEVICE_IS_BLE_SERVER set to 1 in ble_device_type.h.

5. For the code you submit for grading, there should be no calls to __WFI() or __BKPT() or LOG_***() functions. No calls to Systick routines are allowed. However, do log errors returned from API calls.

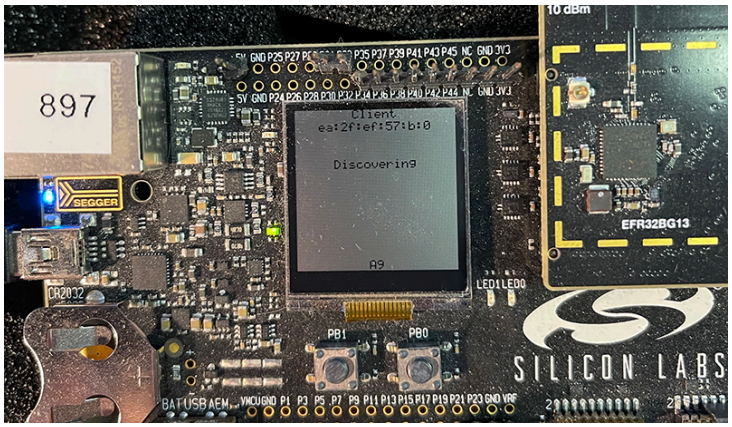
Testing:

Your firmware should pass the following tests:

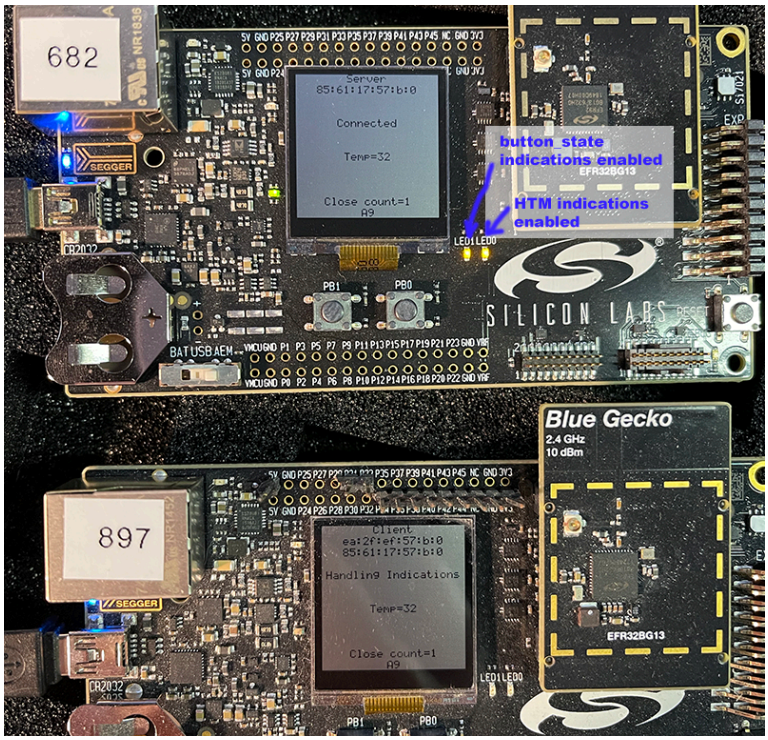
1. All previous behavior from Assignment 7 should be supported regarding the HTM characteristic read and indications, and those tests should continue to pass as written.
2. If not bonded, pressing PB0 on the Server has no effect on the Client
3. Once bonded, with button_state indications enabled:
    a. Pressing and releasing PB0 on your Server shall be reflected on your Client with "Button Pressed" and "Button Released" displayed immediately on row DISPLAY_ROW_9.
4. Once bonded, with button_state indications disabled:
    a. Press and hold PB0 on the Server, then press and release PB1 on the Client. Your Client shall display on DISPLAY_ROW_9 "Button Pressed".
    b. With PB0 released on the Server, pressing and releasing PB1 on the Client shall display on DISPLAY_ROW_9 "Button Released".
5. Powering the Server off should result in the display updating as described above.
6. Powering the Server or Client off and back on should automatically reconnect and restart display updates. It should not be necessary to reset your Client device to restart the sequence.
7. It should be possible to change your Client implementation to access a different Server by changing the value SERVER_BT_ADDRESS in ble_device_type.h.
8. It should be possible to build your working Server A8 implementation by changing the value DEVICE_IS_BLE_SERVER to 1 in ble_device_type.h

Here are pictures of the Client as it sequences through its discovery, connection, bonding and reading of the encrypted characteristic phases.

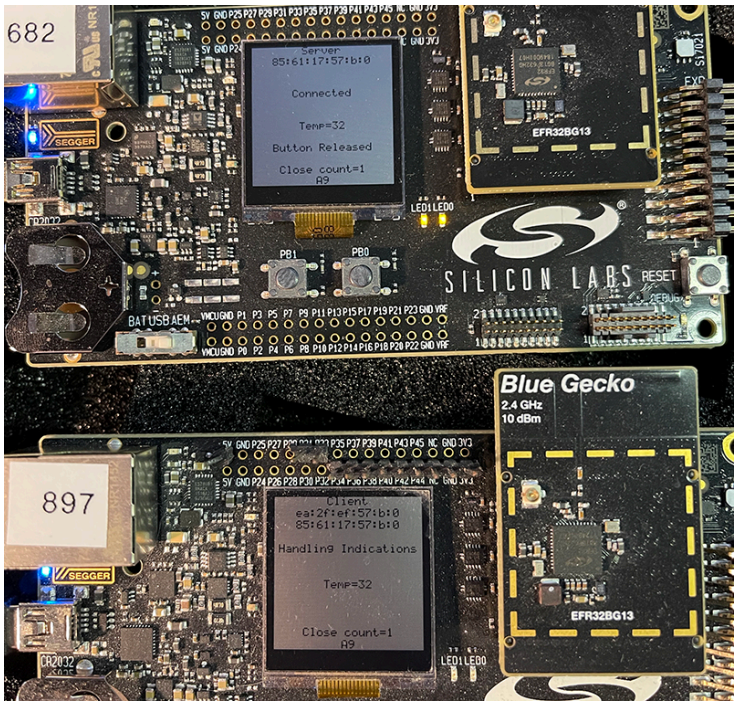
Client discovering services and characteristics on the Sever:



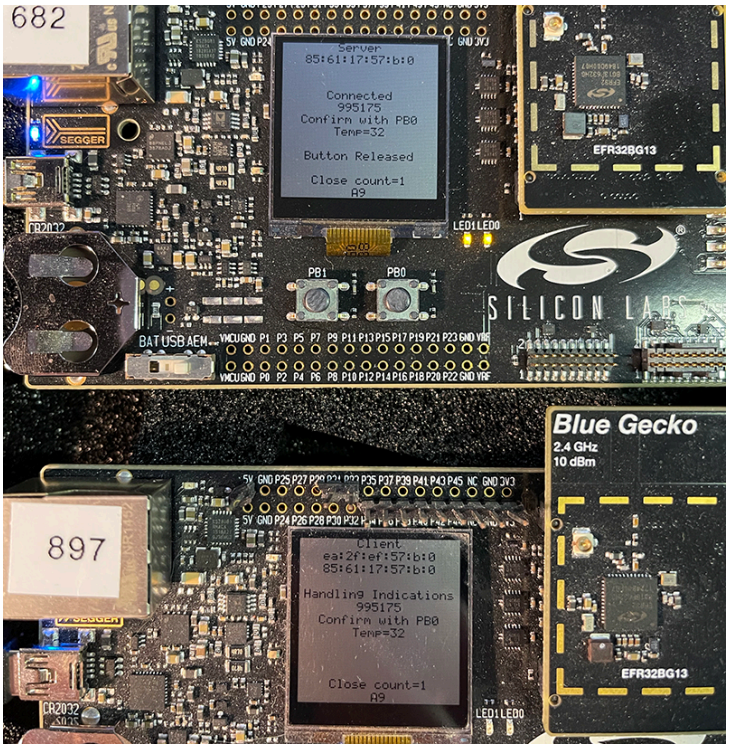
Client and Server after connected and Client has completed discovery and enabled HTM and button_state indications on the Server:



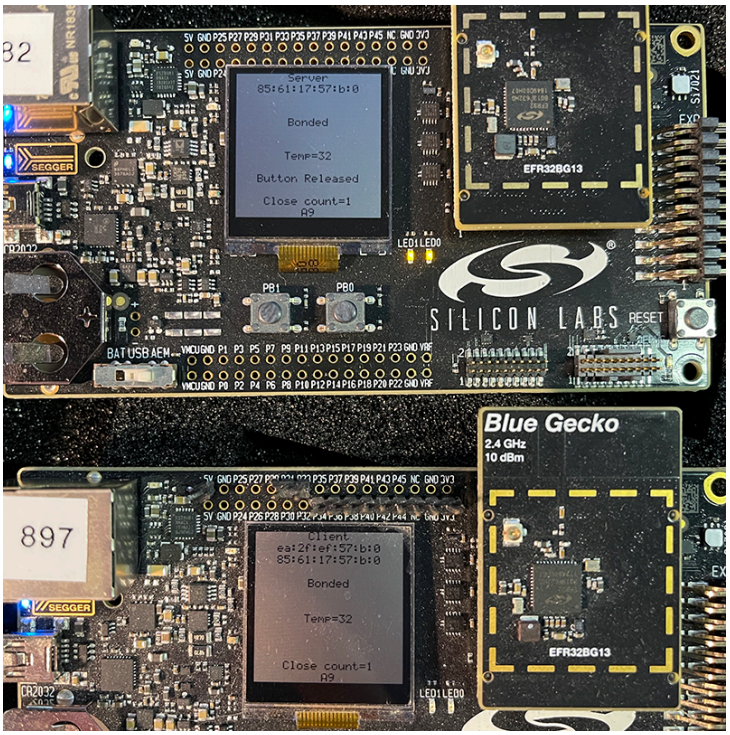
Pressing and releasing PB0 on the Server results in no indications transmitted to the Client because bonding is required for the button_state characteristic indication to be trannsmitted.



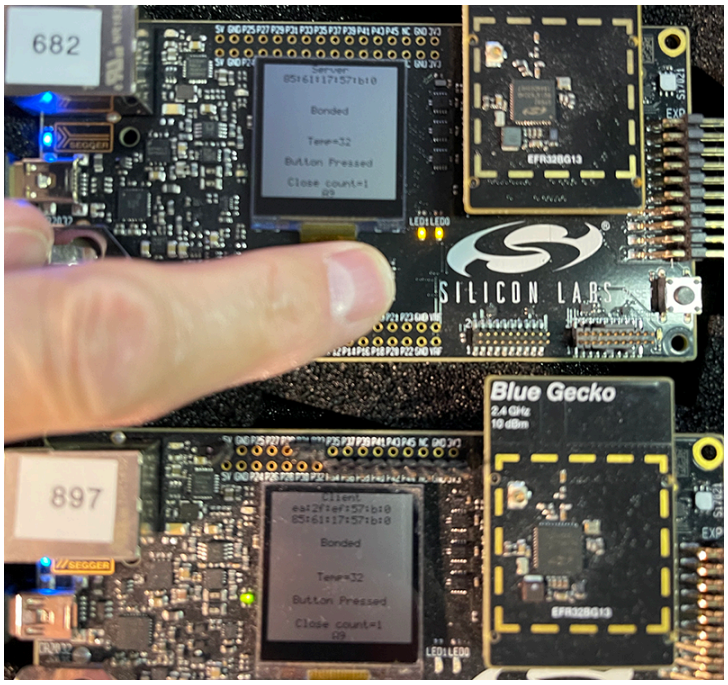
Pressing PB1 on the Client which sends a read characteristic value AP command of the button_state value to the Server, and that AP read fails because an encrypted link is required, and then the failed AP read command triggers the bonding process:



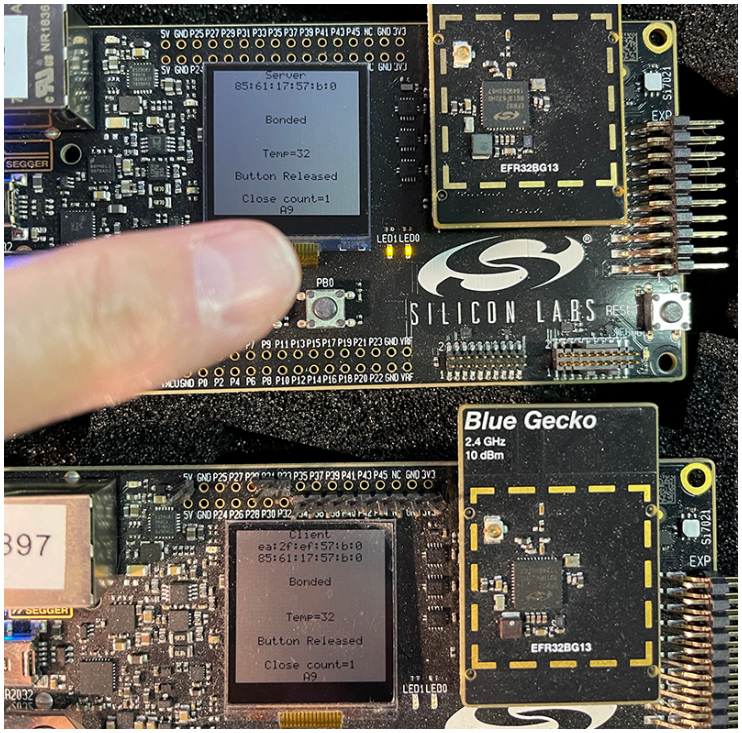
Server and Client after pressing PB0 (Yes) on both devices to confirm bonding:



Button_state indications (Pressed) from the Server to the Client being transmitted when the button_state indications are enabled in the Server:

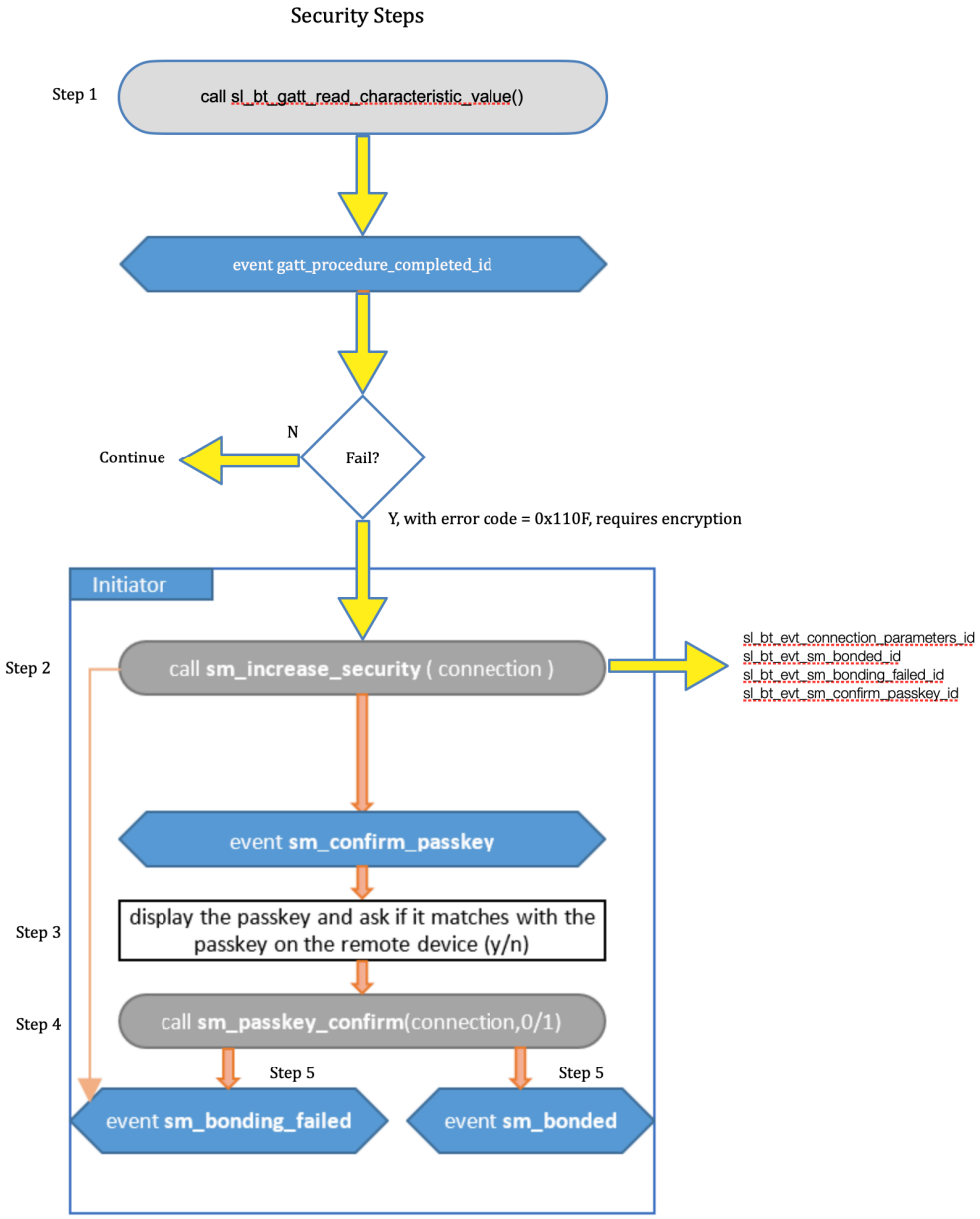


Button_state indications (Released) from the Server to the Client being transmitted when the button_state indications are enabled in the Server:



See the video demo of my A9 in Canvas assignment 9.

As noted above the Silicon Labs flowchart for Numeric Comparison for the Initiator isn't entirely complete. Follow this diagram:



If anyone figures out a way to determine that the button_state attribute requires encryption to read it without first attempting to read the attribute, let me know. This link shows what we can set the attribute permissions to, but the API seems devoid of a method to determine the permissions: https://docs.silabs.com/bluetooth/latest/general/security/using-bluetooth-security-features-in-silicon-labs-bluetooth-sdk#set-up-attribute-permissions-in-gatt

Note: For my own debugging purposes I used the LCD row above the assignment number as a counter of closed events. This is not required for this assignment, however, it can be helpful to display diagnostic information. For the build configuration you submit for grading, please disable all diagnostic LCD messages and logging statements.

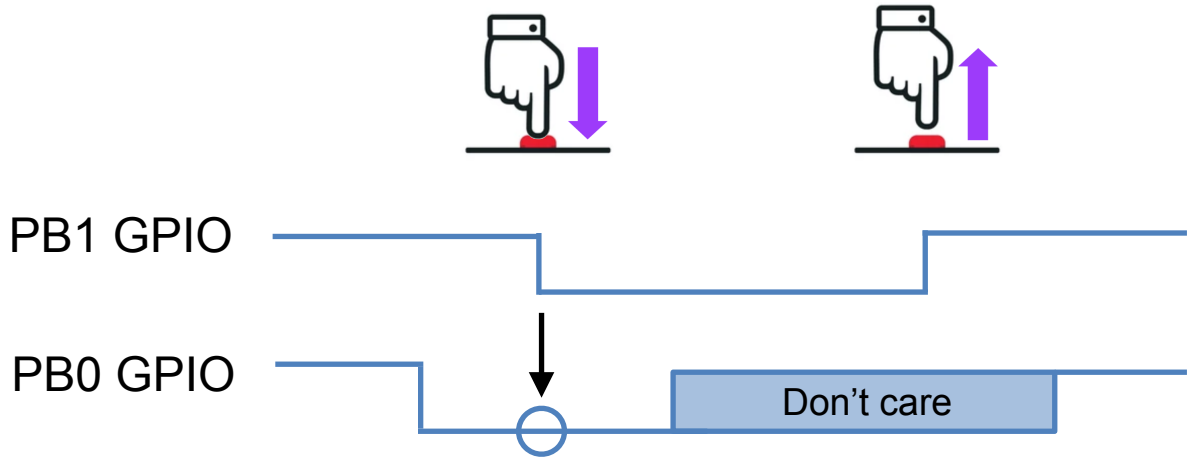
Deliverables:

- Your "Client Command Table A9.xlsx", saved in the "questions" folder of your submission repository. The content should match your A9 Client implementation. Use the git add command to add this new file to your repository.
- The completed assignment 9 program project to be run on the instructing team's computer for grading.
- Submission via **github classroom** and your repository setup via the classroom link at above. Use the following git command to submit your assignment: git push origin master
- Verify your ecen5823-assignment9-<username> repository contains your latest/ intended code and files in the questions folder. It is your responsibility to push the correct version of your project/code. Use a web browser to verify the correct version of your project/code has been uploaded to GitHub.
- In **Canvas**, submit the URL to your github repository and in **GitHub** create a tag of your GitHub submission prior to the due date with the text of A9-final. The date and time of the tag creation is what we use to determine whether your submission was on time or late. These 2 steps will complete your assignment submission.

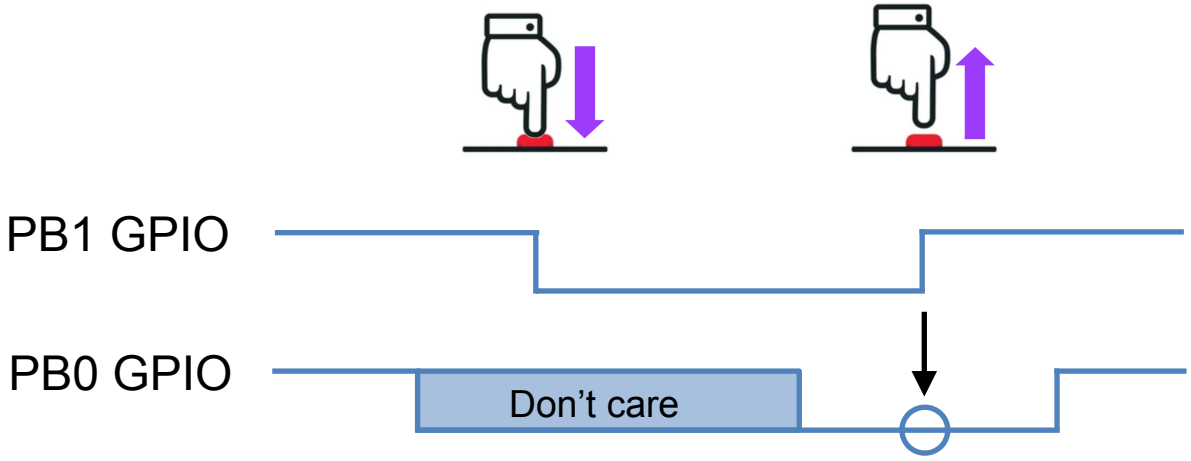
Files in your src/ directory should look like:

Appendix

Button state toggle sequence. The following two figures illustrate the options available to you for detecting PB1/PB0 press and release sequences that will toggle the indication enable for the button_state characteristic in your Server. Make a choice as to which approach you will use.



PB1 falling edge event AND (PB0 == 0)



PB1 rising edge event AND (PB0 == 0)