Matthew Gabriel B619042 Embedded Systems

The task was to create an alarm clock using the Arduino Board. Immediately to me I knew the task would evolve the use of switch statements on button presses and within each case, a function specific to that case would be called to make the code neat and easy to follow. This can be seen as the main approach to the task at hand.

IDLE

The first case is the idle state. This is the default state that is enabled on start up. This counts up seconds and is used as the projects timer. This state calls on the currentTime() function. This function keeps an accurate incrimination of the time, relying on the inbuilt time keeping function millis() and manipulates the value into the 24 hour clock format. This is seen with declaration of seconds, minutes and hours and outputted using sprintf(). Additionally, later on the user can enter the current time and it’s added to the seconds, minutes and hours values. Additionally AM/PM is printed depending on the hours value, less than or greater than 12. Also if the alarm is on, 2 alarm bells are printed and the alarm equals the time the alarm state is triggered. If the select button is pressed, the MENU1 state is entered.

MENU1 & MENU2

This state is a ‘middle man’ for entering the determined menu of choice. Using an if statement combined with the button pressed, changes the state and sends the user to the selected menu option. Also if the screen has been on for longer than 5 seconds, it moves to MENU2 to display the directions on the screen. If MENU2 has been on for more than 10 seconds the state is switched back to IDL. This prevents frustration for the user who may enter the menu states, then enter a period of inactivity, look back at the clock expecting to see the time when the menu state is still active.MENU2 is the same as MENU1 but displays different instructions. I have included “1/2” and “2/2” to indicate that there is another screen to come in the menu states. If select is down then the screen returns to the idle state. One issue that has been reoccurring with the MENU1 and MENU2 states is the inconsistency in the recognition that a button has been pressed and a new state should be entered. This is frustrating for the user as they may have to press a button a few times before the desired state is entered. The source of this problem, I suspect is to do with the timed state change between the two menu options and moving from one states code being run to another in short succession. If I had more time I would look at fixing this problem so that the timed state moves didn’t interfere with the button presses of the user.

SETTIMEONE, SETTIMETWO, SETTIMETHREE

Calls on the setHours(), setMinutes() and setSeconds() functions. All these functions do a similar thing, storing the entered value and adding it to the zeroed millis() value to continue the time from this entered time. The boundary of the entered values are created using if statements, so when the value is decreased/ increased up to a certain point, it is then reset so that the limit is not exceeded. You can navigate between the hours, minutes and seconds by either pressing select after the desired time is entered or using the right and left buttons to move back and forth between hours, minutes and seconds. One issue with this that the navigation between these menus is more implied and assumed that physically displayed on the screen due to lack of space. I opted not to try and squeeze in instructions onto the screen to maintain usability and reduce user frustration at the screen display look.

SETALARMONE, SET ALARMTWO, SET ALARMTHREE

Similar to set time cases, but stores these alarm values for later, to compare to the idle time. Calls on the functions setHours(), setMinutes(),setSeconds() which store the entered alarm values. The same hours, minutes and seconds navigation used in SETIMEONE etc holds in SETALARMONE etc. After the alarm is entered, the user is taken to the TOGGLEALARM state as it is assumed that user once setting an alarm is most likely wanting to turn it on, increasing usability.

DISPLAYALARM

This case is entered from the menu case when the up button is down. This simply calls on the stored initialAlarmHours, initialAlarmMinutes, initialAlarmSeconds values to display the current alarm.

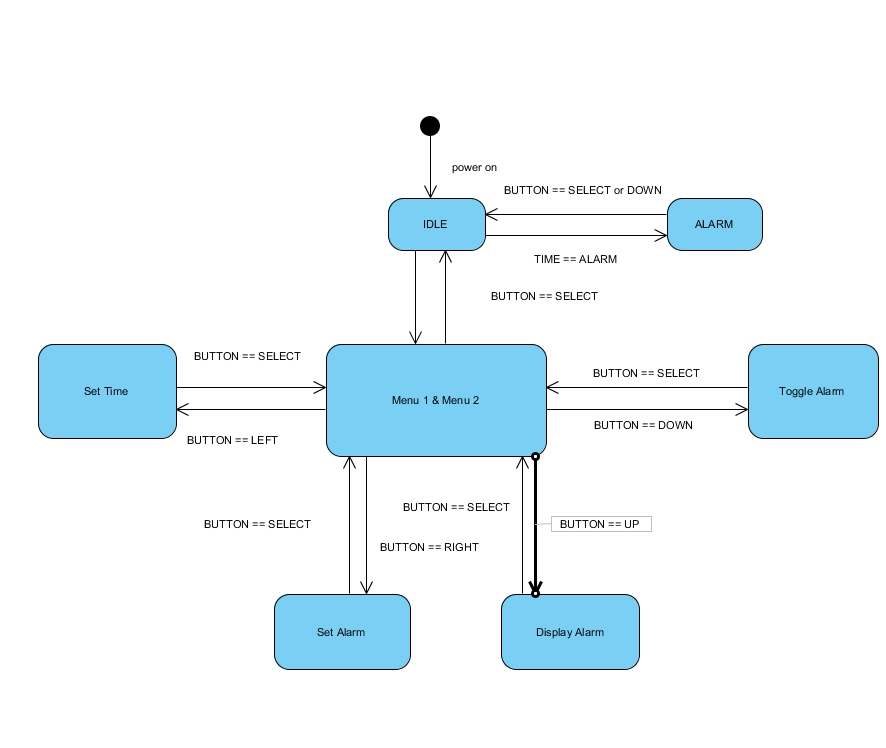
TOGGLEALARM

Toggles the alarm on or off whether down is pressed. After the Alarm is either turned on or off, the state switches back to the IDL state. One recurring issue with this is to do with the timers used for the MENU1 and MENU2 states. Depending on how long the user spends entering the alarm, they may be taken to MENU2 state before going to the desired IDL state. This is something, if I had more time I would look at improving and resetting all the timers so that the states were not confused be timers that are set such as the ones used for the menu states so that they switch accordingly.

ALARM and ALARM2

This state is entered when the entered alarm time is equal to the current time, setting off the alarm state. The background colour quickly flashes between colours and “GET UP ” and instructions on what can be done next flashes between the snooze and stop actions. If down is pressed the snooze mode is entered and 30 seconds is added to the alarm. If select is pressed the IDLE screen is returned to.

Activity Diagram



Tests

|  |  |  |  |
| --- | --- | --- | --- |
| **Description of test** | **Expected Result** | **Actual Result** | **Next Steps** |
| Inputting time into clock 14:23:03 | 14:23:03 displayed when entered | 14:23:03 + millis() since power on is displayed | Edit code to move when the pre-set time is added to millis() so it is added at the right time in the brackets |
| Inputting time into clock 14:23:03 | 14:23:03 displayed when entered | 00:00:00 + millis() displayed | Edit code to move when the pre-set time is added to millis() so it is added at the right time in the brackets |
| Inputting time into clock 14:23:03 | 14:23:03 displayed when entered | 14:23:03 +milis() when the idle state is returned to | No further action |
| Alarm of 15:00:00 is entered and stored | 15:00:00 is stored as the alarm time | 15:00:00 is stored as the alarm time | No further action |
| Alarm ON “ARM” is displayed to indicates that the alarm is on | “ARM” is displayed in right corner | “ARM” is displayed over the time and causes glitching | Edit code to move the lcd.print using lcd.setCursor so it doesn’t overlap |
| Alarm ON “ARM” is displayed to indicates that the alarm is on | “ARM” is displayed in right corner | “ARM” is displayed in the right most corner | No further action |
| When alarm equals the current time the ALARM state is entered | ALARM state entered | ALARM state entered | No further action |
| When down button is pressed during the ALARM state 30 minutes is added to the alarm time | 30 minutes added to alarm value | 30 minutes is added to the alarm time but not capped e.g. 04:90:03 | Add if statement to the code that adds 30 minutes if initialAlarmMinutes=60, set initialAlarmHours=initialAlarmHours+1 and initialAlarmMinutes=0; |
| When down button is pressed during the ALARM state 30 minutes is added to the alarm time | 30 minutes added to alarm value | 30 minutes added to alarm value | No further action |