ENG20009 Portfolio – Practical – Report

Jayden Kong, 104547242

Lab 2 Pass & Pass Plus

Define global variables/constants:

- All button and LED pins
- Button state and button last state variables
- LED state variables, initialise as being either on or off (on = HIGH, off = LOW)
- Time interval for LED 2 blinking, initialise as 1s
- Storing the time of the previous LED 2 blink, initialise as 0
- Brightness level for LED 3, initialise as 0

Setup:

- Set button pins as INPUT_PULLUP and read initial states
- Set LED pins as OUTPUT

Loop:

- Read button states of each button
- If button 1 is pressed:
    - Switch LED 1 on/off
- If button 2 is pressed and current LED 2 blink interval > 0:
    - Decrease LED 2 blink interval by 200ms
- If button 3 is pressed:
    - Increase LED 2 blink interval by 200ms
- If button 4 is pressed and LED 3 brightness < 254:
    - Increase LED 3 Brightness by 20
- If elapsed time is greater than or equal to LED 2 Blink Interval:
    - Update time of previous LED 2 blink
    - Switch LED 2 on/off
- Delay for 10ms

Lab 2 Credit & Distinction

Define global variables/constants:

- Define row and column pins
- Define DIP switch pins and button pin
- Define 8x8 byte matrices for digits 0-9 and letters A-F
- Initialise matrix position variable as 0

Setup:

- Set all row pins as OUTPUT and initialise to HIGH
- Set all column pins as OUTPUT and initialise to LOW
- Set all DIP switch pins as INPUT_PULLUP
- Set the button pin as INPUT_PULLUP

Function scanRow(row):

- Activate the specified row by setting it LOW and the other rows HIGH

Function activateColumn(column, state):

- Set the specified column to the given state (HIGH or LOW)

Function displayCharacter(8x8 character byte matrix)

- Repeat the following five times:
    - For each row in the LED matrix
        - Clear all columns from previous cycle by setting them to HIGH
        - Activate the current row by setting it to LOW
        - For each column in the LED matrix:
            - Determine the scrolled position of the column in the character matrix
            - Activate the current column based on the state of the LED specified by the scrolled character matrix
        - Delay for 2ms
- Scroll the character from left to right if the button is held down (increment position by 1)

Function readDipSwitch():

- Read the state of DIP switches to get a binary value
- Convert the binary value to a base-10 value

Loop:

- Read the state of the DIP switches
- Select the character to be displayed based on the state of the DIP switch
- Display the selected character on the LED matrix

Lab 2 High Distinction

Define global variables/constants:

- Define row and column pins
- Define button pins and buzzer pin
- Button state and button last state variables
- Initialise buzzer on Boolean as false
- Initialise buzzer pitch as 100

Setup:

- Set row and column pins as OUTPUT and initialise them to HIGH
- Set button pins as INPUT_PULLUP
- Set buzzer pin as OUTPUT

Function activateRow(row, state):

- Activate the specified row by setting it to the given state (HIGH or LOW)

Function displayPitch(pitch):

- Display the pitch on the LED matrix display based on the range of values it is between:
    - For each range of pitch values:
        - Activate the rows of the LED matrix that are part of the current pitch level (starting from row 8)
        - Deactivate rows that are not a part of the current pitch level

Loop:

- Read the state of each button
- If button1 is pressed and buzzer is off:
    - Turn on the buzzer with the current pitch
    - Set buzzer on Boolean to true
- If button2 is pressed and buzzer is on:
    - Turn off the buzzer
    - Set buzzer on Boolean to false
- If button3 is pressed, buzzer is on, and pitch is less than 255:
    - Increase the pitch by 30
- If button4 is pressed, buzzer is on, and pitch is greater than 0:
    - Decrease the pitch by 30
- If the buzzer is on:
    - Display the pitch on the LED matrix
- If the buzzer is off:
    - Turn off all rows of the LED matrix
- Delay for 10ms

Lab 3 Pass

Define constants:

- Define LDR pin
- Define LED bar pins

Setup:

- Set LDR pin as INPUT
- Set LED bar pins as OUTPUT and initialise them to LOW

Loop:

- Read the light intensity from the LDR
- If the light intensity is less than 50:
    - Turn on all LEDs
- Else:
    - Turn off all LEDs

Lab 3 Pass Plus

Define constants:

- Define potentiometer pin
- Define LED bar pins

Setup:

- Set potentiometer pin as INPUT
- Set LED bar pins as OUTPUT and initialise them to LOW

Loop:

- Read the analogue value from the potentiometer
- Map the potentiometer value to the range of LEDs, from 0-1023 to 1-11
- For all LED pins:
    - If the LED pin index is less than or equal to the mapped potentiometer value:
        - Turn on the LED
    - Else:
        - Turn off the LED

Lab 3 Credit

Define constants:

- Define potentiometer pin
- Define buzzer pin

Setup:

- Set potentiometer pin as INPUT
- Set buzzer pin as OUTPUT

Loop:

- Read the analogue value from the potentiometer
- Map the potentiometer value from 0-1023 to 1-11
- Set buzzer volume according to the mapped potentiometer value

Lab 3 Distinction

Define global variables/constants:

- Define potentiometer and LDR pin
- Define buzzer and button pin
- Button last state variable
- Button toggled Boolean

Setup:

- Set potentiometer pin as INPUT
- Set LDR pin as INPUT
- Set buzzer pin as OUTPUT
- Set button pin as INPUT_PULLUP
- Initialise button last state as the current state of the button
- Set button toggled Boolean to false

Function tone1(light intensity, potentiometer value):

- Play a tune with variations based on light intensity and the potentiometer value

Function tone2(light intensity, potentiometer value):

- Play another tune with variations based on light intensity and the potentiometer value

Loop:

- Read the analogue value from the potentiometer
- Read the light intensity from the LDR
- Map the light intensity from 0-1023 to 0-255
- Read the state of the button
- If the button is pressed:
    - Switch button toggled Boolean as true/false

Lab 3 Distinction loop continued ->

- If button toggled Boolean is true:
    - Play tone 2
- Else:
    - Play tone 1

Lab 3 High Distinction

Define global variables/constants:

- Define row and column pins
- Define button pin
- Define potentiometer pin
- Initialise matrix position variable as 0
- Initialise waveform selection variable as 0 (sine wave)
- button last state variable
- Define 8x16 byte matrices for waveforms sine, triangle, square and sawtooth

Setup:

- Set button pin as INPUT_PULLUP
- Set potentiometer pin as INPUT
- Set row and column pins as OUTPUT
- Initialise button last state to the current state of the button

Function scanRow(row):

- Activate the specified row by setting it HIGH and the other rows LOW

Function activateColumn(column, state):

- Set the specified column to the reverse of the given state (HIGH or LOW)

Function displayWave(8x16 waveform byte matrix, frequency):

- Repeat the following for as many times as specified by the given frequency:
    - For each row in the LED matrix
        - Activate the current row by setting it to HIGH
        - For each column in the LED matrix:
            - Determine the scrolled position of the column in the waveform matrix
            - Activate the current column based on the state of the LED specified by the scrolled waveform matrix
        - Delay for 1ms
- Scroll the waveform from left to right (increment position by 1)

Loop:

- Read the analogue value from the potentiometer
- Map the potentiometer value to a frequency value between 1 and 10
- Read the state of the button

Lab 3 High Distinction loop continued ->

- If the button is pressed:
    - Switch to the next waveform
- Based on the currently selected waveform:
    - Display the selected waveform on the LED matrix

Lab 4 Pass and Pass Plus

Define global variables/constants:

- Define LED pins
- LED state variables, initialise as being either on or off (on = HIGH, off = LOW)
- Time interval for LED 2 blinking, initialise as 1s
- Storing the time of the previous LED 2 blink, initialise as 0
- Brightness level for LED 3, initialise as 0

Setup:

- Set baud rate for serial data transmission at 9600
- Set LED pins as OUTPUT

Function cycleLEDs():

- Switch LED 1 on/off
- If elapsed time is greater than or equal to LED 2 Blink Interval:
    - Update time of previous LED 2 blink
    - Switch LED 2 on/off
- Set LED 3 to its current brightness level

Function confirmationMenu():

- Infinite loop until function is told to return:
    - Print out confirmation menu
    - While waiting for user input:
        - Perform regular LED cycle specified by function cycleLEDs()
    - Read user entered integer as option selection:
        - If user entered 1 (yes):
            - Return true
        - If user entered 2 (no):
            - Return false
        - If user entered none of the above options
            - Print invalid option message

Function LED1ToggleMenu():

- while user is not finished with menu:
    - Print out LED 1 toggle menu
    - While waiting for user input:
        - Perform regular LED cycle specified by function cycleLEDs()
    - Read user entered integer as option selection:
        - If user entered 1 (toggle LED 1):
            - If confirmed wanted:
                - Switch LED 1 on/off
        - If user entered 2 (return to main menu):
            - Finish with menu
        - If user entered none of the above options
            - Print invalid option message

Function LED2IncreaseMenu():

- while user is not finished with menu:
    - Print out LED 2 blink speed increase menu
    - While waiting for user input:
        - Perform regular LED cycle specified by function cycleLEDs()
    - Read user entered integer as option selection:
        - If user entered 1 (increase LED 2 blink speed):
            - If confirmed wanted:
                - Decrease LED 2 blink interval by 250ms
        - If user entered 2 (go to LED 2 blink speed decrease menu):
            - Finish with menu, go to LED 2 blink speed decrease menu
        - If user entered 3 (return to main menu):
            - Finish with menu
        - If user entered none of the above options
            - Print invalid option message

Function LED2DecreaseMenu():

- while user is not finished with menu:
    - Print out LED 2 blink speed decrease menu
    - While waiting for user input:
        - Perform regular LED cycle specified by function cycleLEDs()
    - Read user entered integer as option selection:
        - If user entered 1 (decrease LED 2 blink speed):
            - If confirmed wanted:
                - Increase LED 2 blink interval by 250ms
        - If user entered 2 (go to LED 2 blink speed increase menu):
            - Finish with menu, go to LED 2 blink speed increase menu
        - If user entered 3 (return to main menu):
            - Finish with menu
        - If user entered none of the above options
            - Print invalid option message

Function LED3BrightnessMenu():

- while user is not finished with menu:
    - Print out LED 3 brightness control menu
    - While waiting for user input:
        - Perform regular LED cycle specified by function cycleLEDs()
    - Read user entered integer as option selection:
        - If user entered 1 (increase LED 3 brightness):
            - If confirmed wanted and LED 3 brightness < 254:
                - Increase LED 3 brightness by 60
        - If user entered 2 (decrease LED 3 brightness):
            - If confirmed wanted and LED 3 brightness > 0:
                - Decrease LED 3 brightness by 60
        - If user entered 3 (return to main menu):
            - Finish with menu
        - If user entered none of the above options
            - Print invalid option message

Loop:

- Print out main menu
- While waiting for user input:
    - Perform regular LED cycle specified by function cycleLEDs()
- Read user entered integer as option selection:
    - If user entered 1 (Toggle LED 1):
        - Go to LED 1 toggle menu
    - If user entered 2 (Increase LED 2 blink speed):
        - Go to LED 2 increase menu
    - If user entered 3 (Decrease LED 2 blink speed):
        - Go to LED 2 decrease menu
    - If user entered 4 (Control LED 3 brightness):
        - Go to LED 3 brightness menu
    - If user entered none of the above options
        - Print invalid option message

Lab 4 Credit

Initialisation:

- Define everything required for the RTC and TFT display

Setup:

- Begin communication with the RTC
- Adjust the RTC time to the time the sketch was uploaded
- Initialise the TFT display with the appropriate settings:
    - Landscape orientation
    - White text colour and text size 3
    - Disable text wrapping
- Clear the TFT display

Loop:

- Read the current time from the RTC
- Format the current time into the format hh:mm:ss
- Display formatted time on the TFT display
- Delay for 1s
- Clear TFT display

Lab 4 Distinction

Initialisation:

- Define everything required for the IMU and TFT display

Setup:

- Initialise the IMU
- Initialise the TFT display with the appropriate settings:
    - Landscape orientation
- Clear the TFT display

Loop:

- Obtain accelerometer data from the IMU
- If the x-axis acceleration is positive:
    - Print left arrow
- If the x-axis acceleration is negative:
    - Print right arrow
- If the y-axis acceleration is positive:
    - Print down arrow
- If the y-axis acceleration is negative:
    - Print up arrow
- Delay for 100ms
- Clear the TFT display

Lab 4 High Distinction

Initialisation:

- Define everything required for the IMU and TFT display

Define global variables/constants:

- Current position of symbol (symbolX, symbolY)
- Symbol speed (xspeed, yspeed)
- Initial accelerometer readings (accelInitX, accelinitY)

Setup:

- Set baud rate for serial data transmission at 115200
- Initialise the IMU
- Initialise the TFT display with the appropriate settings:
    - Landscape orientation
    - White text colour and text size 2
    - Disable text wrapping
- Obtain initial accelerometer readings from the IMU

Loop:

- Obtain accelerometer data from the IMU
- Calculate the symbol speed based on the difference between current and initial accelerometer readings
- Update the position of the symbol on the TFT display by adding the symbol speed to the current position of the symbol
- If the symbol is at the border of the TFT display:
    - Stop the symbol from going offscreen
- Print the symbol position to the serial monitor
- Print the symbol at the symbol position on the TFT display
- Delay for 100ms
- Clear the TFT display

Lab 5 Pass and Pass Plus

Initialisation:

- Initialise everything required for the TFT display
- Initialise everything required to use PROGMEM

Store strings in PROGMEM:

- Student ID and name
- Unit name and semester

Setup:

- Initialise the TFT display with the appropriate settings:
    - Landscape orientation
    - White text colour and text size 3
    - Disable text wrapping
- Clear TFT Display

Function printText(PROGMEM array):

- For every character stored in the PROGMEM array:
    - Print the character to the TFT display
    - If the character read is the null terminator:
        - Stop printing text

Function textLength(PROGMEM array):

- Initialise a count variable as 0
- For every character stored in the PROGMEM array:
    - Increment the count variable by 1
    - If the character read is the null terminator:
        - Stop reading text
- Return the final count as the length of the text
-

Function scrollText(PROGMEM array):

- Set the text to be initially offscreen of the right border of the TFT display
- Calculate the length offset for the text stored in the PROGMEM array
- Loop until the text is completely offscreen:
    - Decrement the x-coordinate to scroll the text from right to left
    - Print the text stored in PROGMEM at the current position
    - Delay for 10ms
    - Clear the TFT display

Loop:

- print and scroll student ID
- print and scroll name
- print and scroll unit name
- print and scroll semester

Lab 5 Credit

Initialisation:

- Initialise everything required for the TFT display
- Initialise everything required to use EEPROM

Setup:

- Set baud rate for serial data transmission at 9600
- Initialise the TFT display with the appropriate settings:
    - Landscape orientation
    - White text colour and text size 3
    - Disable text wrapping
- Clear the TFT display
- Find the address of EEPROM and print the address to the terminal

Function searchFunction():

- Search for address of DS28E07 chip
- Print 8 addresses found to the serial monitor

Function writeToEEPROM(user input string):

- Convert input string to a char array
- Write the char array to EEPROM byte by byte

Function readFromEEPROM(number of EEPROM addresses):

- Initialise an array to store data bytes read from EEPROM
- Loop through each byte to read from EEPROM and store data that is read in the array
- Convert array of data bytes to a string and remove the first 3 bytes
- Return the string made from the data bytes array

Loop:

- Write first 7 characters of student ID to EEPROM
- Read 8 bytes stored in EEPROM
- Remove characters beyond the 7^{th} index from the string that was read in
- Print the modified string on the TFT display
- Write last 2 characters of student ID to EEPROM
- Read 8 bytes stored in EEPROM
- Remove characters beyond the 2^{nd} index from the string that was read in
- Print the modified string on the TFT display
- Delay loop for 1s
- Clear TFT display

Lab 5 Distinction

Initialisation:

- Initialise everything required for the RTC, SD card and TFT display

Define global variables/constants:

- Define button pins
- Button state and button last state variables
- Time interval for clock screen refresh, initialise as 1s
- Storing the time of the previous clock screen refresh, initialise as 0
- Use 24-hour time format Boolean, initialise as true
- Alarm time variables, initialise as 00:00:00
- Alarm active Boolean, initialise as false
- Alarm triggered Boolean, initialise as false
- Display flash Boolean, initialise as false

Setup function:

- Set baud rate for serial data transmission at 9600
- Begin communication with the RTC
- Adjust the RTC time to the time the sketch was uploaded
- Initialise the TFT display with the appropriate settings:
    - Landscape orientation
    - White text colour
    - Disable text wrapping
- Clear the TFT display
- Set button pins as INPUT_PULLUP
- Setup SD card
- Remove button input log file if it exists

Function clockMenu():

- Clear TFT display
- Set text size to 1
- While not finished with clock settings menu:
    - Print out clock settings menu on TFT display
    - Read button states of each button
    - If button 1 is pressed:
        - Switch use 24-hour time format Boolean between true/false
        - Write clock information to SD card
    - If button 2 is pressed:
        - Write clock information to SD card
        - Go to set time menu
    - If button 3 is pressed:
        - Write clock information to SD card
        - Go to set alarm menu
    - If button 4 is pressed:
        - Write clock information to SD card
        - Finish with clock settings menu

Function setTimeMenu():

- Clear TFT display
- Initialise new time variables as 00:00:00
- While not finished with set time menu:
    - Print new time to be set and set time menu options
    - Read button states of each button
    - If button 1 is pressed:
        - Go to next time field
        - Write clock information to SD card
    - If button 2 is pressed:
        - Increase current time field by 1
        - Write clock information to SD card
    - If button 3 is pressed:
        - Adjust the RTC to the new time
        - Write clock information to SD card
        - Finish with set time menu
    - If button 4 is pressed:
        - Write clock information to SD card
        - Finish with set time menu

Function setAlarmMenu():

- Clear TFT display
- Initialise new alarm time to be set using the current alarm time
- While not finished with set alarm menu:
    - Print new alarm time to be set and set alarm menu options
    - Read button states of each button
    - If button 1 is pressed:
        - Go to next alarm time field (seconds cannot be selected)
        - Write clock information to SD card
    - If button 2 is pressed:
        - Increase current alarm time field by 1
        - Write clock information to SD card
    - If button 3 is pressed:
        - Adjust the alarm time to the new alarm time
        - Set alarm active Boolean to true
        - Write clock information to SD card
        - Finish with set alarm menu
    - If button 4 is pressed:
        - Adjust the alarm time to the new alarm time
        - Write clock information to SD card
        - Finish with set alarm menu

Function clockInformation():

- Create a char array to store information about the clock
- Store current time, alarm time, alarm active Boolean, and use 24-hour time format Boolean in the char array
- Return the char array

Function WriteSD(button input log file, clock information string):

- Open the button input log file in write mode
- Print the clock information string to the end of the file
- Close the file

Loop:

- Read button states buttons 1 and 3
- If button 1 is pressed:
    - Write clock information to SD card
    - Go to clock settings menu
- If button 3 is pressed and alarm has been triggered:
    - Disable colour inversion on the TFT display (if screen is in flashed state, return to normal)
    - Set alarm active and alarm triggered Booleans as false
    - Write clock information to SD card
- If elapsed time is greater than or equal to clock screen refresh Interval:
    - Clear TFT display
    - Read the current time from the RTC
    - If alarm hours and minutes is equal to the current hours and minutes:
        - Set alarm triggered Boolean as true
    - Print out current time in the time format specified by the use 24-hour time format Boolean
    - If alarm has been triggered:
        - Enable/disable colour inversion on the TFT display

Lab 5 High Distinction

Initialisation:

- Initialise everything required for the RTC, IMU, SD card and TFT display

Define global variables/constants:

- Arrays to store accelerometer x, y and z data
- Current index variable, initialise as 0

Setup:

- Set baud rate for serial data transmission at 115200
- Begin communication with the RTC
- Adjust the RTC time to 00:00:00
- Initialise the TFT display with the appropriate settings:
    - Landscape orientation
    - Text size 1
    - Disable text wrapping
- Clear the TFT display
- Initialise the IMU
- Setup SD card

Lab 5 High Distinction setup continued ->

- Remove IMU data log file if it exists

Function drawGraphLine(data array, line colour):

- Obtain the current index
- Set graph starting x-coordinate at 60
- Starting from the current index to the beginning of the data array:
    - Draw lines connecting each data point to the previous one

Function concatenateSensorData(accelerometer x, y, z, time data was obtained):

- Create a char array to store sensor data
- Store time data was obtained, and accelerometer x, y and z values in the char array
- Return the char array

Function WriteSD(IMU data log file, sensor data string):

- Open the IMU data log file in write mode
- Print the sensor data string to the end of the file
- Close the file

Loop:

- Get current time from RTC
- Obtain accelerometer data from the IMU
- Map accelerometer data to the y-coordinate range of the display
- Print out legend for the graph line colours
- Draw graph lines for x, y, and z data on the TFT display
- Write sensor data to the SD card
- Delay for 1s
- Clear the TFT display
- Increase current index by 1

Lab 6 Pass

Initialisation:

- Initialise everything required for the TFT display

Define constants:

- Define button pin

Setup:

- Initialise the TFT display with the appropriate settings:
    - Landscape orientation
    - White text colour, text size 3
    - Disable text wrapping
- Set button pin as INPUT_PULLUP
- Attach interrupt to the button pin to trigger function displaySymbol on button press
- Clear the TFT display

Function displaySymbol():

- Clear the TFT display
- Print symbol at a random location on the TFT display

Loop: Not used

Lab 6 Pass Plus

Initialisation:

- Initialise everything required to use timer interrupts on the Arduino Due

Define global variables/constants:

- Define LED pin
- LED state variable, initialise as LOW

Setup:

- Set LED pin as OUTPUT
- Attach function toggleLED() to Timer3 interrupt
- Start Timer3 with a period of 3.5s to turn on the LED

Function toggleLed():

- Switch the LED on/off
- If LED is off
    - Restart Timer3 with a period of 3.5s
- Else:
    - Restart Timer3 with a period of 1s

Loop: Not used

Lab 6 Credit

Initialisation:

- Initialise everything required for the IMU and TFT display
- Initialise everything required to use timer interrupts on the Arduino Due

Define global variables/constants:

- Arrays to store x, y and z data
- Current index variable, initialise as 0
- Current graph variable, initialise as 1 (magnetometer)
- Define button pins
- Button state and button last state variables

Setup:

- Initialise the TFT display with the appropriate settings:
    - Landscape orientation
    - Text size 1
- Clear the TFT display
- Initialise the IMU
- Set button pins as INPUT_PULLUP
- Initialise button last state variables as the current state of the buttons
- Attach Timer3 interrupt to function getSensorData with a frequency of 0.5Hz

Function getSensorData():

- Obtain accelerometer data from the IMU
- Read current graph variable as graph selection:
    - If current graph variable is 1:
        - Store magnetometer data in x, y and z arrays
    - If current graph variable is 2:
        - Store gyroscope data in x, y and z arrays
    - If current graph variable is not any of the above:
        - Store accelerometer data in x, y and z arrays
- Increase current index by 1

Function drawGraphLine():

- Obtain the current index
- Set graph starting x-coordinate at 60
- Starting from the current index to the beginning of the data array:
    - Draw lines connecting each data point to the previous one

Loop function:

- Read button states of each button
- If button 1 is pressed and current graph variable is not 1:
    - Set current graph variable to 1 (magnetometer)
    - Reset the current index variable to 0
- If button 2 is pressed and current graph variable is not 2:
    - Set current graph variable to 2 (gyroscope)

Lab 6 Credit loop continued ->

    - Reset the current index variable to 0
- If button 3 is pressed and current graph variable is not 3:
    - Set current graph variable to 3 (accelerometer)
    - Reset the current index variable to 0
- Print out legend for the graph line colours
- Draw graph lines for x, y, and z data on the TFT display
- Delay for 1s
- Clear the TFT display

Lab 6 Distinction

Initialisation:

- Initialise everything required for the TFT display
- Initialise everything required to use timer interrupts on the Arduino Due

Define global variables/constants:

- Define button pin
- Time variables, initialise as 00:00:00
- Digital clock style Boolean, initialise as false (hardware interrupt in setup will change this to true)
- Constants for the analogue clock:
    - Centre coordinates of TFT display
    - Length of hours, minutes and seconds hands

Setup:

- Set button pin as INPUT_PULLUP
- Change time as specified to something arbitrary (for example, 16:42:00)
- Initialise the TFT display with the appropriate settings:
    - Landscape orientation
    - White text colour, text size 2
    - Disable text wrapping
- Attach interrupt to the button pin to trigger function changeClock on button press
- Attach Timer3 interrupt to function incrementClock with a frequency of 1Hz
- Clear the TFT display

Function changeClock():

- Switch digital clock style Boolean between true/false

Function incrementClock():

- Increment seconds
- update minutes and hours accordingly

Function displayAnalogue():

- Calculate angles for hour, minute, and second hands based on current time
- Draw clock hands using the calculated angles and specified hand lengths
- Draw a circle to represent the clock face and a smaller filled circle at the centre

Loop:

- If digital clock style Boolean is true:
    - Display the current time in the format hh:mm:ss
    - Print out formatted time to the TFT display
- Else:
    - Display the analogue clock
- Delay for 1s
- Clear the TFT Display

Lab 6 High Distinction

Define global variables/constants:

- Define row and column pins
- Define 8x8 byte matrices for letters A-Z and a space
- Initialise matrix position variable as 0
- Define button pin
- Scroll flip direction Boolean, initialise as false
- Scroll speed, initialise as 10 (slow)
- Button press time, initialise as 0
- User input text char array
- Concatenated character matrices matrix
- Text total width variable, initialise as 0

Setup:

- Set all row and column pins as OUTPUT
- Set baud rate for serial data transmission at 9600
- Get user input text from serial monitor
- Concatenate character matrices required to print user input text
- Set button pin as INPUT_PULLUP
- Attach interrupt to the button pin to trigger function buttonIsPressed() on button press and release
- Set scroll speed as 10 (slow)

Function getText():

- While not finished getting user input:
    - Wait for user to enter a string into serial monitor
    - Copy user input string into user input text char array
    - Make all alphabet characters in array lowercase
    - Initialise a valid string Boolean as true
    - If input text is not made up of only alphabet characters and spaces:

Lab 6 High Distinction getText() continued ->

        - Set valid string Boolean as false
    - If valid string Boolean is true:
        - Finish with user input
    - Else:
        - Print invalid input message to serial monitor
    - If input text is greater than 100 characters:
        - Do not finish with user input
        - Print invalid input message to serial monitor

Function concatenateMatrices():

- Set all elements of the concatenated character matrices matrix as 0
- Initialise a column offset as 0
- For all characters in the user input text array:
    - Copy all character array contents to the concatenated matrix according to the column offset
    - Increase column offset by 8
    - Increase text total width by 8
- Increase text total width by 8 (creates space at the end of the matrix)

Function buttonIsPressed():

- Read current state of button
- If button is not pressed down:
    - Calculate button press time
    - If button press time is less than 375ms:
        - Switch scroll flip direction Boolean between true/false
    - Else:
        - Switch to next scroll speed (slow to medium, medium to fast, fast to slow)
- Else:
    - Start calculating button press time
- Update Last interrupt time

Function scanRow(row):

- Activate the specified row by setting it HIGH and the other rows LOW

Function activateColumn(column, state):

- Set the specified column to the reverse of the given state (HIGH or LOW)

Loop:

- Repeat the following for as many times as specified by scroll speed:
    - For each row in the LED matrix
        - Activate the current row by setting it to HIGH
        - For each column in the LED matrix:
            - Determine the scrolled position of the column in the concatenated character matrix
            - Activate the current column based on the state of the LED specified by the concatenated character matrix

Lab 6 High Distinction loop continued ->

        - Delay for 2ms
- If scrolling direction is reversed (right to left)
    - Decrement the matrix position by 1 (scroll right to left)
- Else:
    - Increment the matrix position by 1 (scroll left to right)