ENG 20009 - Engineering Technology Inquiry Project Semester 1 2023

Practical Report

Student Name / ID : Joshua Lillington-Moore / 103666887
Tutor and Time : Khanh Ha Nguyen, Thursday 12.30-2.30pm

Table Of Contents :

Lab 2 :

Pass

- Using 4 push buttons to control the behavior of 3 LEDs on the bar graph or buzzer:
    - The first push button should toggle the first LED/buzzer on and off.

https://wokwi.com/projects/359953807060675585 - debounce

https://wokwi.com/projects/359700884279174145

```
> No Debounce
setup() {
//OUTPUT for LED Bar
pinMode(pinNumber (eg. 8) , OUTPUT);
//INPUT for Push Buttons
pinMode(pinNumber (eg.1), INPUT);
}

loop() {
  // read the value for the Push Buttons
  button_value = digitalRead(1);

  if (button_value is HIGH (been pressed)) {
          Turn corresponding LED Bar pin on
  }
  else
          Turn corresponding LED Bar pin off

>Debounce
//define Push Button and LED Pins
buttonPin = 1;
ledPin = 8;
ledPin2 = 9;
ledPin3 = 10;

//setup Pin Modes for the LED and Push Buttons
ledState = LOW;
ledState2 = LOW;
ledState3 = LOW;
buttonState;
lastButtonState = LOW;

//define debounce time
lastDebounceTime = 0;
```

```
//define debounce delay, suitable time before button can be pushed again
debounceDelay = 50;

setup() {
define the pins for the LEDs (OUTPUT)
define the pins for the push buttons (INPUT)

//Initial LED States to corresponding pins
  digitalWrite(ledPin, ledState);
}

loop() {
Read the button pin values (if they have been pressed or not)
{
//check if the button was pressed
  if (reading != lastButtonState) {
    lastDebounceTime = millis(); // sets timer when button is debounced
  }
/*if time between millis() and lastDebounceTime is greater than the delay of the debounce (50)
than enough time has passed to run through the loop*\
  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != ButtonState) {
      buttonState = reading;
//if button has been pressed, switch values of ledState
    if (buttonState is equal to HIGH) {
        ledState = !ledState;
        ledState2 = !ledState2;
        ledState3 = !ledState3;
      }
    }
  }
}
//Update LED states and Last Button states before going through Loop again
  digitalWrite(ledPin, ledState);
  lastButtonState = reading;
}
```

Pass +

- Continue from Pass question above with the following tasks:
    - The second button should increase the speed of the second LED's blinking.
    - The third button should decrease the speed of the second LED's blinking.
    - The fourth button should toggle the brightness of the third LED between high and low.

https://wokwi.com/projects/359711641003526145

```
//define pins
const buttonPin4 = 4;
const ledPin3 = 10;
const brightStep = 5; //each time it loops how much brightness it adds
const delayTime = 100;
brightness = 0; //initial brightness level

setup() {
ledPins set to OUTPUT and buttons set to INPUT
set led 3 to off at start
}

loop() {
 //get initial values for buttons
  int button_value = value of corresponding pin;

//button1
  if (button is pressed ){
      turn LED for pin 1 on;
  }
  else
      turn LED for pin 1 off;

 //button2, makes it flash every 500ms
 if (button2 has been pressed){
      turn LED on pin 9 on, wait 500ms, than turn LED on pin 9 off, wait 500ms again
}
 //button3, slower, flashes every 2000ms
 if (button3 has been pressed){
      turn LED on pin 9 on, wait 2000ms, than turn LED on pin 9 off, wait 2000ms again
}
//button4, LED brightness increases and decreases
  if (button 4 has been pressed ){
    for (i = brightness; i <= 255; i+=brightStep){ //gradually increases the LED brightness
      brightness = i;   /*let the brightness equal to i, so it increases by the brightstep each time
through the loop*/
      set ledPin3 equal to the value of brightness
      delay(delayTime); //waits before starting to decrease
    }
    for (int i = brightness; i >= 0; i-=brightStep){ //gradually decreases the LED brightness
      brightness = i;
      analogWrite(ledPin3, brightness);
      delay(delayTime);
    }
```

```
//wait for button to be released before continuing loop
   delay(100);
   while (digitalRead(buttonPin4) == LOW){
     delay(10);
   }
  }
}
```

Lab 3 :

Pass

- Create menu using the UART and display in the serial monitor which has selection at the following:
    - The first menu should toggle the first buzzer on and off

https://wokwi.com/projects/360537532800158721

```
define ledpin and buadRate for serial monitor

setup() {
set ledpin as OUTPUT
start serial monitor with baud rate
//create menu
print("------Menu to Control LED------");
print("Enter Option :");
print("1 : LED On");
print("2 : LED Off");
}

loop() {
  if (Serial.available() > 0){      //if serial is on
    int command = Serial.read(); //sets command to user input
    if (command == '1'){
      set ledPin to high
      println("LED on");
    }else if (command == '2'){
      set ledPin to low
      Serial.println("LED off");
    }else{ //if input is unknown
      print("Unknown Command: ");
      print(command);
    }
  }
}
```

Pass +

- Continue from the Pass question above with more selection below:
    - The second menu should increase the speed of the second LED's blinking.
    - The third menu should decrease the speed of the second LED's blinking.
    - The fourth menu should toggle the brightness of the third LED between high and low.

https://wokwi.com/projects/360549513324972033

```
//define pins
buzzerPin = 8;
ledPin = 9;
brightStep = 5; //each time it loops how much brightness it adds
delayTime = 100;
//set initial values for blink and fadeLED functions
int blinkInterval = 1000;
unsigned long lastBlinkTime = 0;
int brightness = 0;

setup() {
  define buzzerPin  and ledPin as OUTPUT
  Start serial monitor and print menu with the 4 options
}

void loop() {
  // put your main code here, to run repeatedly:
  if (Serial is on){
    char input = user input into the serial;
    switch (input) {
      case '1':
          tone(8, 350, 1000); //makes the buzzer go off
          print("Buzzer On");
          break;
      case '2':
          if (blinkInterval < 200) { //calls function
            blinkInterval -= 200; //decreases blink interval
          }
          print("LED Blinking Increased");
          break;
      case '3':
            blinkInterval += 200;  //increase blink interval
          print("LED Blinking Decreased");
          break;
      case '4':
```

```
          fadeLed(); //calls function
        default:
          ledPin set to LOW
          break;
    }
  }
  blink(); //blinks led to whatever it has been set in the function
}

blink() { //blink function
  currentMillis = millis(); //stores millis() value
  if (currentMillis - lastBlinkTime >= blinkInterval) {
    lastBlinkTime = currentMillis; //update lastBlinkTime
    toggle ledState
}
}

fadeLed(){ /fadeLED function
    for (int i = brightness; i <= 255; i+=brightStep){ //led brightness increases by brightStep
      brightness = i;
      analogWrite(ledPin, brightness);
      delay(delayTime);
    }
    for (int i = brightness; i >= 0; i-=brightStep){  //led brightness decreases by brightStep
      brightness = i;
      set ledPin equal to the value of brightness
      delay(delayTime);
    }
    delay(100);
}
```

Credit

- Using the RTC and LCD. Create a digital clock by reading the data from RTC and displaying it on LCD.

https://wokwi.com/projects/360617812055495681

```
include the libraries for the RTC and SSD1306

define screen width and height
initialize SSD1306 display
initialize the RTC

setup() {
 // Initialize the I2C bus
```

```
 Wire.begin();

 rtc.begin(); //start RTC
//can manually set time or leave as comment to get current time
//rtc.adjust(DateTime(YYYY, MM, DD, hh, mm, ss));
//rtc.adjust(DateTime(0,0,0,15,50,10));

// Initialize the OLED display
// Clear the display
}

loop() {
  // Clear the display
  display.clearDisplay();
 //set the display TextColor, Size and Cursor Location

 DateTime now = rtc.now(); //use inbuilt function to get current time
 display the time on the LCD
//use in built function for rtc library, now.hour, now.minute, now.second
}
```

Distinction

- Using the accelerometer of the IMU create a spirit level that displays the current angle away from level on the LCD display and provide a graphic that will assist with leveling the board.

https://wokwi.com/projects/363127968245782529

```
include libraries for ssd1306, accelerometer and wire connection
define screen width and height
initialize SSD1306 display
initialize the MPU6050 (accelerometer and gyroscope)
//define min and max values for the gyroscope, used to find the angle
minVal=265;
maxVal=402;
//define the variable for the angles, x,y and z

setup() {:
  Start serial
//check if MPU6050 is connected
  while (!mpu.begin()) {
    print("MPU6050 not connected!");
    delay(1000);
  }
```

```
  print("MPU6050 ready!");
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_5_HZ);
  // Initialize the OLED display
  display.display();
  delay(500); // Pause for 2 seconds
  Set display TextSize,Color and Rotation
}

void loop() {
read accelerometer and gyro values
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  clearDisplay() and setCursor(0, 0);
//map gyro readings (g.gyro.x etc) into angles, using the min and max Values
  int xAng = map(g.gyro.x,minVal,maxVal,-90,90);
  int yAng = map(g.gyro.y,minVal,maxVal,-90,90);
  int zAng = map(g.gyro.z,minVal,maxVal,-90,90);
//calculate the angles using atan2 function and convert into degrees
  x= RAD_TO_DEG * (atan2(-yAng, -zAng)+PI);
  y= RAD_TO_DEG * (atan2(-xAng, -zAng)+PI);
  z= RAD_TO_DEG * (atan2(-yAng, -xAng)+PI);

/*print and display the values of the accelerometer, gyroscope and the angles calculated
using in built function, a.acceleration and g.gyro*/
Serial.print("Accelerometer ");
  print("X: ");
  print(a.acceleration.x, 1);
  display("Accelerometer - m/s^2");
  display(a.acceleration.x, 1);
  print("Gyroscope ");
  print("X: ");
  print(g.gyro.x, 1);
  display.display();
  delay(1000);
//print out angles into serial monitor
  print x,y,z
}
```

Lab 4 :

Pass

- Create a night-activated LDR sensor (use Analog instead of Digital) to turn on the LED bar during night time and turn off LED bar during day time.

https://wokwi.com/projects/361333894051074049

```
define LED and LDR sensor pins

setup() {
set ledPins to OUTPUT and ldrPin to INPUT
}

loop() {
  if (LDR is set to LOW (below 100)){
    Led is turned off
    delay(500);
  }
  else {
    Led turned on
    delay(500);
  }
}
```

Pass +

- Instead of LDR, use potentiometer to adjust the brightness of the LED bar

https://wokwi.com/projects/361335050555996161

```
Define pins for LEDs and Potentiometer
//potentiometer must use analog pins

setup() {
set ledPins to OUTPUT
//LED Bar does not adjust brightness, thus can use an LED in order to show it
}

loop() {
  potentValue = analogRead(potentPin);
//map the brightness of the LED to the potentiometer value
  brightness = map(potentValue, 0, 1023, 0, 255);
  analogWrite(ledPins, brightness);
  analogWrite(ledPin, brightness);
```

```
}
```

Credit

- Write a program that controls the volume of noise/tone from the speaker/buzzer using input from the potentiometer.

https://wokwi.com/projects/361344517561919489

```
Define pins for Buzzer and Potentiometer
//potentiometer must use analog pins

setup() {
Set buzzerPin to OUTPUT
}
//noise is the frequency
loop() {
//map the value of potentiometer to the frequency of the buzzerPin
  int potentValue = analogRead(potentPin);
  int noise = map(potentValue, 0, 1023, 0, 255);
  tone(buzzerPin, noise);
}
```

Distinction

- Distinction: Using a microphone to create a simple access control. The passcode/password is “123”. This passcode needs to be activated via voice over the mic. If the passcode is entered correctly, shows graphical symbol or image for correct authentication, otherwise shows incorrect image in the display (SSD1306 OLED Display or TFT-LCD display

https://wokwi.com/projects/361610277918295041

```
//include the required libraries for the SSD1306 and program library
include <Wire.h>,  <Adafruit_GFX.h>, <Adafruit_SSD1306.h>, <avr/pgmspace.h>

define screen width and height
//initialize the SSD1306
//bitmaps from https://javl.github.io/image2cpp/
//create arrays in Program Memory for the bitmaps
nuclearError [] PROGMEM = {};
accessGranted [] PROGMEM = {};

void setup() {

Initialize the SSD1306, display, delay and than clear its display
}
```

```
void loop() {
  clearDisplay();
  //set the display TextColor, Size and Cursor Location
  Print and display ("Enter Passcode : ");

  while (Serial.available() == 0){}
  String input =readString(); //reads user input
  input.trim(); //removes blank spaces, otherwise it wont recognize it
  print(input); //take user input

  if (input == "123"){
      clearDisplay();
      print("Authentication Complete");
      display.drawBitmap(0, 0,  accessGranted, 128, 64, WHITE);
      display.display();
      delay(2000);
      exit(0); //exit program
    } else {
      clearDisplay();
      print("Incorrect, try again");
      display.drawBitmap(0, 0,  nuclearError, 128, 64, WHITE);
    }
    display.display();
    delay(2000);
    clearDisplay();
}
```

Lab 5 :

Pass & Pass +

- Store the following list in PROGEM, then print them from PROGEM onto the LCD screen (SSD1306 OLED Display or TFT-LCD display). Each item from the list should scroll from right to left. - *Student ID* - *Student name*
- Continue from the Pass question above with following list: - ENG20009 - Engineering Technology Inquiry Project - Semester 1 - 2023

https://wokwi.com/projects/361607221154343937

```
//include the required libraries for the SSD1306 and program library
include <Wire.h>,  <Adafruit_GFX.h>, <Adafruit_SSD1306.h>, <avr/pgmspace.h>

define screen width and height
//initialize the SSD1306
```

```
//store data in the program memory using PROGMEM
studentID[] PROGMEM = "103666887";
name[] PROGMEM = "Joshua Lillington";
unitCode[] PROGMEM = "ENG20009";
unitName[] PROGMEM = "Engineering Technology Inquiry Project";
semester[] PROGMEM = "Semester 1";
year[] PROGMEM = "2023";
//create array consisting of all data
const char *const allData[] PROGMEM = {studentID, name, unitCode, unitName, semester,
year};
//buffer to store the data
buffer[50];

setup() {
  initialize the serial monitor
  while (!Serial);

  initialize the SSD1306 display
  display();
  delay(2000);
  clearDisplay();
}

loop() {
  clearDisplay();
  set the display TextColor, Size and Cursor Location
  display.setTextWrap(false); //makes it so it doesn't overlap text on the display

//loop through the data, display it one line at a time
  for (int i = 0; i < 6; i++)
  {
    strcpy_P(buffer, (char *)pgm_read_word(&(allData[i])));  //copy data from progmem to buffer
    display(buffer);
    display.startscrollleft(0, 7); //scrolls all text on display left, built in function
    println(buffer);
    delay(500);
  }

  display();
  delay(2000);
  clearDisplay();
}
```

Credit

- Connect to the EEPROM component. Write to the component your student ID and display it back on the LCD (SSD1306 OLED Display or TFT-LCD display) from the EEPROM.

https://wokwi.com/projects/362494258373826561

```
//include the required libraries for the SSD1306 and EEPROM
include <Wire.h>,  <Adafruit_GFX.h>, <Adafruit_SSD1306.h>, <EEPROM.h>

define screen width and height
initialize the SSD1306

//create array for studentID
studentID[20] = "103666887";

setup() {
  initialize the serial monitor
  while (!Serial);

  initialize the SSD1306
  display() delay(2000) and than clearDisplay();
//loop to write the studentID into the EEPROM
  int i;
  for (i = 0; i < sizeof(studentID); i++) //gets the size of the array
  {
    EEPROM.write(i, studentID[i]); //writes each character one at a time
  }
  print(studentID);
}

loop() {
  clearDisplay();
  set the display TextColor, Size and Cursor Location
  display(studentID);
  display.display();
}
```

Lab 6 :

Pass

- Using the interrupt hardware for pushbutton, display a non-alphanumeric symbol on the LCD

https://wokwi.com/projects/363049001477054465

```
Include libraries for SSD1306
Define screen width and height
Initialize the SSD1306 display

buttonPin = 2;
buttonState = 0;  //value will change

void setup() {
  Set buttonPin to input
/*use attachInterrupt to create an external interrupt for the button, the interrupt handler will
CHANGE the state of the button when it is pressed*/
  attachInterrupt(digitalPinToInterrupt(buttonPin), buttonInterruptHandler, CHANGE);
  Start SSD1306 and display
}

loop() {
  if (buttonState) { //if button is pressed
    buttonState = 0; //set buttonState back to 0
    Clear display and set the text size and color
//set the cursor to the middle of the screen
    display.setCursor((display.width() - 12) / 2, (display.height() - 16) / 2);
    display on OLED the ASCII Value
    delay(1000);
  }
}
//create function for the InterruptHandler
buttonInterruptHandler() { //when button is pressed and released buttonState is set to HIGH
  buttonState = 1;
}
```

Pass +

- Continue Pass question above, using the interrupt hardware for pushbutton, display various symbols on the LCD, each time the button is pressed it should trigger an interrupt to change the symbol.

https://wokwi.com/projects/363052192524777473

```
Include libraries for SSD1306
Define screen width and height
Initialize the SSD1306 display

buttonPin = 2;
buttonState = 0;  //value will change
```

```
count = 0; //how many times the button is pressed
//create an array containing non-alphanumeric symbols
symbols[] = {234, 157, 153, 219};

void setup() {
  Set buttonPin to input
/*use attachInterrupt to create an external interrupt for the button, the interrupt handler will
CHANGE the state of the button when it is pressed*/
  attachInterrupt(digitalPinToInterrupt(buttonPin), buttonInterrupt, CHANGE);
  Start SSD1306 and display
}

void loop() {
  if (buttonState) {
    buttonState = 0;
    Count++; //add to count each time it is pressed
    if (count >= sizeof(symbols)) {  //sizeof = amount in array
      count = 0; // wraps around when it reaches the end of the array
    }
    clear display and set the text size and color
    set the cursor to the middle of the screen;
    display on the OLED the current symbol depending on the value of count
    display.write(symbols[count]);
    update display
    delay for 1 second
  }
}

buttonInterrupt() {
  buttonState = 1; //set buttonState to HIGH
}
```

Credit

https://wokwi.com/projects/363061869947834369

```
Include libraries for SSD1306
Define screen width and height
Initialize the SSD1306 display
//variable for seconds minutes and hours
seconds, minutes, hours;  //values will change

buttonPin = 2;
buttonState = 0; //value will change
```

```
//adjust values manually here, have to comment above as well
//volatile int seconds = 50;
//volatile int minutes = 2;
//volatile int hours = 1;

setup() {:
  Set buttonPin to input
/*use attachInterrupt to create an external interrupt for the button, the interrupt handler will
CHANGE the state of the button when it is pressed*/
  attachInterrupt(digitalPinToInterrupt(buttonPin), buttonInterrupt, CHANGE);
  Start SSD1306 and display

  Update display and than clear display
  Set the displays text size, color and cursor position

  stop interrupts
//setup internal timer interrupt, may vary depending on the device
  TCCR1A = 0;// set entire TCCR1A register to 0
  TCCR1B = 0;// same for TCCR1B
  TCNT1  = 0;//initialize counter value to 0
  // set compare match register for 1hz increments
  OCR1A = 15624;// = (16*10^6) / (1*1024) - 1 (must be <65536)
  // turn on CTC mode (counter cleared to 0)
  TCCR1B |= (1 << WGM12);
  // Set CS12 and CS10 bits for 1024 prescaler
  TCCR1B |= (1 << CS12) | (1 << CS10);
  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);

  allow interrupts
  clearDisplay();
}

ISR(TIMER1_COMPA_vect){ //setup interrupt service routine
   seconds ++; //increment second counter
   if(seconds == 60)
   {
   seconds = 0; //reset sec counter
   minutes ++; //increment min counter
//Update minute counter
   if(minutes == 60)
   {
   minutes = 0; //reset min counter
   hours ++; //increment hr counter
```

```
//Update hour counter
    if(hours == 24)
    {
    hours = 0; //reset hour counter
    }
  }
 }
 if (hours < 10) {
    Serial.print("0"); // add leading zero for hours less than 10
  }
  Serial.print(hours);
  Serial.print(":");
//do this for minutes and seconds also to print in serial monitor

loop() {
    Clear the display and set the cursor
//display the time on the OLED
    if (hours < 10) {
    display.print("0"); // add leading zero for hours less than 10
  }
    display.print(hours);
    display.print(":");
  if (minutes < 10) {
    display.print("0"); // add leading zero for minutes less than 10
  }
    display.print(minutes);
    display.print(":");
  if (seconds < 10) {
    display.print("0"); // add leading zero for seconds less than 10
  }
  display.print(seconds);
  display.display();
}
void buttonInterrupt() {
  buttonState = 1; //changes buttonState to high
}
```