

SB Micros Assignment

by Sarah Butler

Submission date: 14-Jan-2025 09:27AM (UTC+0000)

Submission ID: 248375879

File name: 51378_Sarah_Butler_SB_Micros_Assignment_621295_1393924934.docx (561.12K)

Word count: 2805

Character count: 15191

Using and programming the 7-segment display prototype circuit

ACDA4004 – Micros, Peripherals and Interfaces

Sarah Butler

2315177

Table of Contents

SECTION 1: 4

SECTION 2: 7

SECTION 3: 10

 TASK 1: 10

 TASK 2: 13

 TASK 3: 16

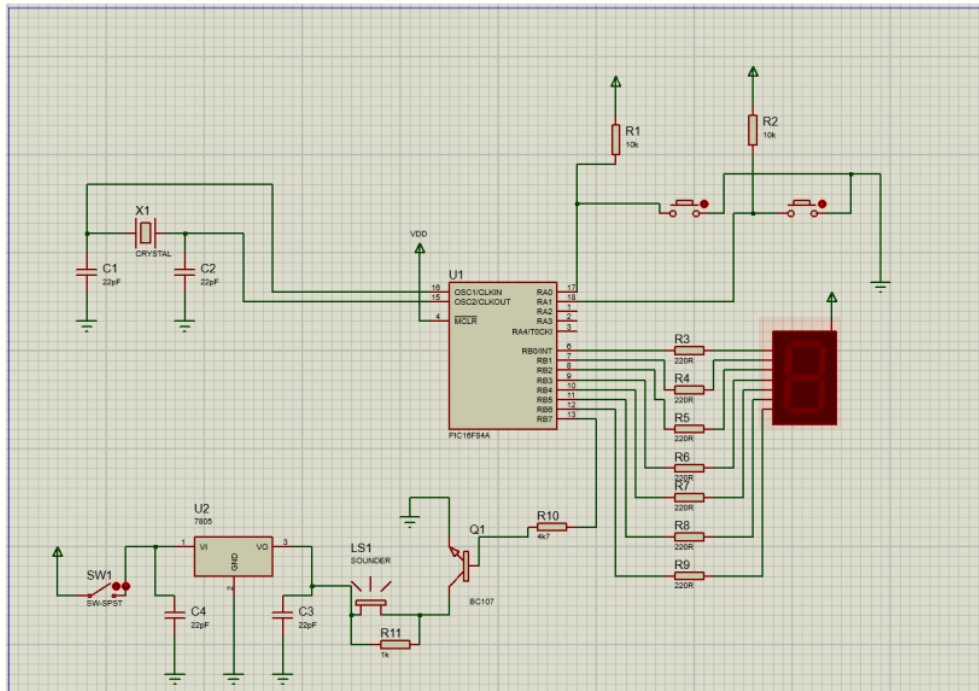
References 17

FIGURE 1: DIAGRAM OF LABELLED 7-SEGMENT DISPLAY 4

FIGURE 2: CAPACITOR IN SERIES WITH CELL [7].....5

FIGURE 3: SILICON NPN LOW POWER BIPOLAR TRANSISTOR [8].....	5
FIGURE 4: MOMENTARY SPST PUSH BUTTON [10]	5
FIGURE 5: QUARTZ CRYSTAL AS USED IN A CIRCUIT.....	6
FIGURE 6: RESISTOR IN SERIES WITH BATTERY.....	6
FIGURE 7: PIC16F84A WITH LABELLED PINS	6
FIGURE 8: INSIDE OF PIEZO SOUNDER.....	6
FIGURE 9: SPST SWITCH [9]	6

SECTION 1:



This is a Proteus Simulation created to match the original for this project. Listed below are the components that make the circuit and their purpose, along with data and diagrams for explanation.

- 7-Segment Display, common anode – when 0 lights up
 - The 7 LEDs of this display are arranged in a way that forms a rectangle which can form numerical digits.
 - Each of the LEDs are positioned to be representative of a part of a digit and are labelled 'a'-'g'.
 - This design of the 7-Segment display uses a common anode. This means all LEDs are joined at logic 1, when applying a ground to each segment, that segment of LED turns on [1]

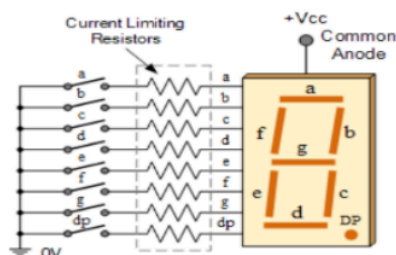


Figure 1: Diagram of labelled 7-Segment Display

- Capacitor

- The capacitor stores electrical energy between two conductive plates.
- In this circuit, capacitors are used for filtering, to smooth out fluctuations in voltages.

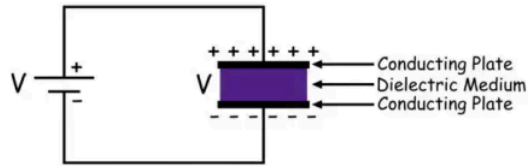


Figure 2: Capacitor in series with Cell [7]

- Silicon NPN low power bipolar transistor

- The NPN bipolar transistor is a semiconductor that can amplify or act like a switch for electrical signals.
- It takes a small electrical current input at the base which controls a larger current between the collector and emitter, which is can amplify or switch.

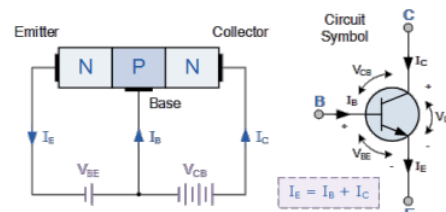


Figure 3: Silicon NPN low power bipolar transistor [8]

- 5V power supply regulator

- Takes an input voltage often between 35V - 7V and outputs a consistent 5V. This protects components from fluctuations [2].
- The power supply regulator in this circuit protects components from fluctuation of input voltage by producing a consistent 5 volts from the higher voltage supply.

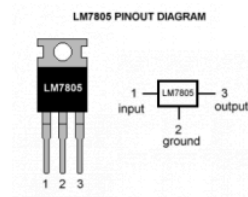


Figure 4: Positive Power supply Regulator

- SPST push button

- SPST means single pole single throw – which refers to this type of button having one input contact and one output contact.
- The function of the button in this circuit is as an option for operation of a different task.

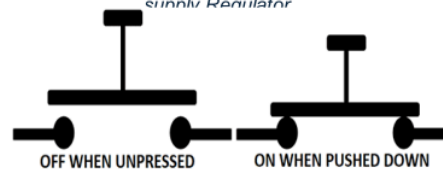


Figure 4: Momentary SPST Push Button [10]

- Quartz crystal
 - o Quartz crystal is a piezo electric device used to create precise frequencies [3].
 - o It is used for the clock on the microcontroller. It produces precise frequencies which allows the microcontroller stable clock signals.

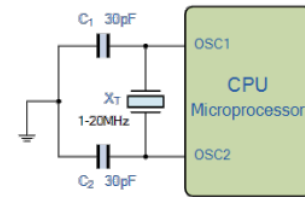


Figure 5: Quartz Crystal as used in a circuit

- Resistors
 - o Resistors are used to regulate or limit the rate of electrical flow [4]
 - o In this circuit, they are used to protect the LEDs of voltages which cause damage and generally regulate voltage.



Figure 6: Resistor in series with battery

- PIC16F84A microcontroller
 - o A microcontroller controls electronic devices and manages inputs and outputs and executes programmed instructions.
 - o The PIC16F84A is typically used for small tasks such as smaller machinery, managing small toy functions and processing data collected by sensors [5]. In this case, it is being programmed to function the 7-Segment display. The segments of the display, a-g, are assigned a binary number through the pins of the microcontroller – starting at 1 for 'a', 2 for 'b', 4 for 'c', and so on to 64 for 'g'. This is how they are communicated with to programme.

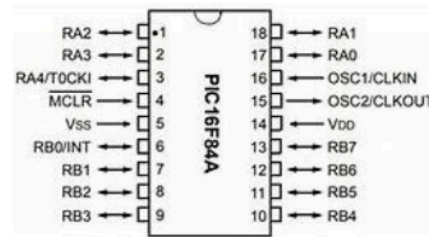


Figure 7: PIC16F84A with labelled pins

- Piezo sounder
 - o The piezo sounder is an electronic device that produces sound when an alternating voltage disturbs the dielectric material [6]
 - o In this circuit it is used as one of the tasks to programme the PIC16 to control.

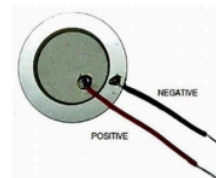


Figure 8: Inside of Piezo sounder

- SPST switch
 - o The SPST switch is a switch to allow or stop electrical flow, which has one input contact and one output contact.

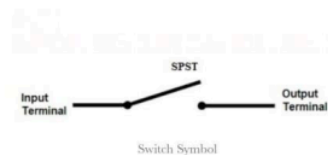


Figure 9: SPST Switch [9]

SECTION 2:

The circuit created in Proteus simulation allows all Hexadecimal digits to be displayed on the 7-Segment Display.

To display all characters, the first step to take is designing how the Hexadecimal characters should appear on the display – to understand what to work towards. Producing the characters is done by using the graph of the datasheet for the 7-Segment display; each segment of the display needs to be turned off or on which creates the characters. As the display being used is a common anode type, the pin output of the microcontroller – which is assigned to the individual segments which the programme works with, needs to be turned off for the light to be on.

For example, to work out how to display the first digits of the display, 0, the centre, horizontal segment needs to be turned off and all other segments on. Using the design graph of the 7-Segment Display being used, the LED that needs to be off is segment G. The binary for this design of character is simply 64, as only one segment needs to be turned off; however, if additional segments would need to be turned off like in other hexadecimal characters, the sum of the binary representations would be used. With the binary representation of the LED assignment, this is used to find the hexadecimal value of the LED assignment – which is used in the code. For this case, the hexadecimal value is 40.

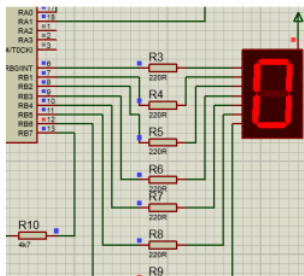


Figure 10: Pic output pin status'

As shown in the image beside, pins B0-B5 are blue meaning they are off, and the LED segments are on. B6 pin is red meaning that it is on and the horizontal LED segment it is aligned with is off.

Hexadecimal Character	Segments Needed	Off	Bitwise Calculation	Binary Value	Hex Value
0	g		64	64	0x40
1	a,d,e,f,g		$1+8+16+32+64$	121	0x79
2	c,f		$4+32$	36	0x24
3	e,f		$16+32$	48	0x30
4	a,d,e		$1+8+16$	25	0x19
5	b,e		$2+16$	18	0x12
6	b		2	2	0x02
7	d,e,f,g		$8+16+32+64$	120	0x78
8	none		0	0	0x00
9	e		32	32	0x18
A	d		16	16	0x08
B	a,b		$1+2$	3	0x03
C	b,c,g		$2+4+64$	70	0x46
D	a,f		$1+32$	33	0x21
E	b,c		$2+4$	6	0x06
F	b,c,d		$2+4+8$	14	0x0E

Figure 11: Table of Bitwise Calculations

display on the LEDs.

All information for each Hexadecimal character is in Figure 11. It shows the character to be recreated, the segments of the display that need to be turned off (which is the information important to the code which makes this happen), the calculation made of this information, and the conversions which allow this data to be input into the code and

To set up the code to receive the values of the hexadecimal characters, a few things need to be put in order.

First the variables and definitions need to be set up that are later called on in the program. For this, it was defining register on the microchip and setting up counter variable loops.

Then, starting the circuit, the first step is to clear the bit selection in the status register, preparing the code. The next step is to use the 7-Segment Display. This is done by selecting the bank it's on (bank 0), and loading it with '0x00' to calibrate it ready for the sequence. After loading that into trisB it makes portB all outputs which link to the display.

The next step is to clear all lines low and initialize the counter at zero to start the sequence of characters to follow.

To display the Hexadecimal characters, the value calculated before needs to be called. For each character that is called, it follows the same code to display it.

The function is created to keep the code of the display for the characters under the name 'characters'. This is so it can be called upon to display when programmed to. Within the function, the code to display the first number is written as so.

```
characters
    movlw 0x40 ;0
    movwf portB
    call pause
```

Figure 12: Example of how '0' is displayed on the 7-Segment Display

The 'movlw' command means to move 'l', the literal value of 0x40 into 'w', the working register. 'movwf' command, then means that the value just moved to the working register is to be stored in the location 'portB'. 'pause' is another function made within the program which delays – in this example its used to display the character on the 7-segment display for a desired amount of time. The time is set within the pause function and for this code it is 200ms. 'call pause' means that it has been used wherever called so the code above delays for 200ms.

```
characters
    movlw 0x40 ;0 - moves the literal of 0x40 into working register
    movwf portB ; moves the value in working register into location portB
    call pause ; delay for 200ms
    movlw 0x79 ;1 - moves the literal of 0x79 into working register
    movwf portB ; moves the value in working register into location portB
    call pause ; delay for 200ms
    movlw 0x24 ;2 - moves the literal of 0x24 into working register
    movwf portB ; moves the value in working register into location portB
    call pause ; delay for 200ms
```

Figure 13: Part of the characters function in code showing the continuation into the next character

This is then repeated for all Hexadecimal characters, 0-F. By using this format, the code runs line after line and displays each character similarly.

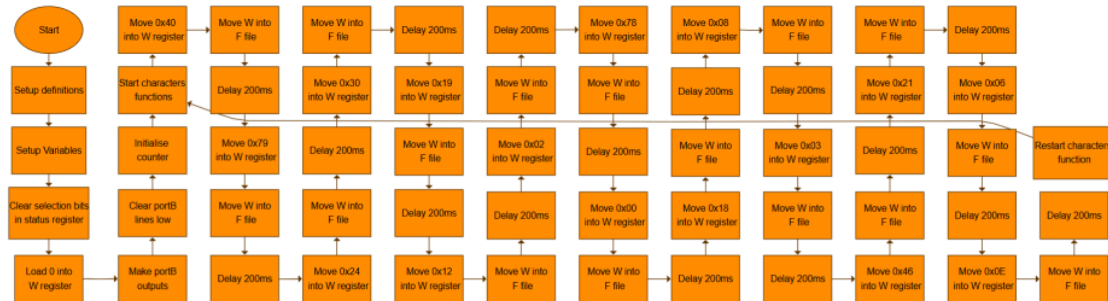
When the code reaches the last character, F, there is one last line which restarts the process. By using 'goto characters' the active code line is sent to the start of the 'characters' function which starts by displaying '0'. This makes the function a continuous loop.

```
movlw 0x0E ;F - moves the literal of 0x79 into working register
movwf portB ; moves the value in working register into location portB
call pause ; delay for 200ms
goto characters
```

Figure 14: End of the function

SECTION 3:

TASK 1:



Start

```
    bcf    status,RP0    ; clear selection bits in the status reg
    bcf    status,RP1

    bsf    status,RP0    ; reselect bank 0
    movlw  0x00          ; load w with 0
    movwf  trisB          ; load it into trisB to make portB all outputs
    bcf    status,RP0    ; reselect bank 0

    clrf   portB          ; clear all lines low
    clrf   count          ; initialize counter to 0
```

characters

```
    movlw  0x40 ;0 - moves the literal of 0x40 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x79 ;1 - moves the literal of 0x79 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x24 ;2 - moves the literal of 0x24 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x30 ;3 - moves the literal of 0x30 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x19 ;4 - moves the literal of 0x19 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x12 ;5 - moves the literal of 0x12 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x02 ;6 - moves the literal of 0x02 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x78 ;7 - moves the literal of 0x78 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x00 ;8 - moves the literal of 0x00 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x18 ;9 - moves the literal of 0x18 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x08 ;A - moves the literal of 0x08 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x03 ;B - moves the literal of 0x79 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x46 ;C - moves the literal of 0x79 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x21 ;D - moves the literal of 0x79 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x06 ;E - moves the literal of 0x79 into working register
    movwf  portB ; moves the value in working register into location portB
    call   pause ; delay for 200ms
    movlw  0x0E ;F - moves the literal of 0x79 into working register
```

```

    movwf portB ; moves the value in working register into location portB
    call  pause ; delay for 200ms
    goto  characters

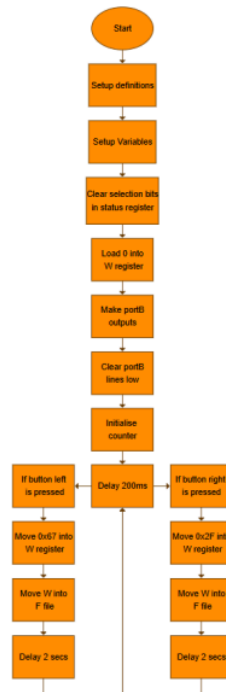
get_count
    movf  count,w      ; move count to W
    goto  characters
; movwf portB          ; move W to portB to update the display
; addlw  1             ; add +1 to W
    movwf count        ; save W in count
;=====

pause
    movlw 0xff         ; set W = 255 decimal
    movwf mCount       ; mCount = w: outer loop counter
loadN
    movlw 0xff         ; set w = 255 decimal
    movwf nCount       ; nCount = w: inner loop counter
decN
    decfsz nCount, f    ; nCount--
    goto  decN          ; if nCount != 0 then repeat nCount--

    decfsz mCount, f    ; else decrement Count
    goto  loadN         ; if mCount != 0 then
                        ; reload nCount to 255 and decrement
    return             ; else exit subroutine
;=====
END

```

TASK 2:



```

;=====
; Main.asm file generated by New Project wizard
;
; Created:   Wed Dec 4 2024
; Processor: PIC16F84A
; Compiler:  MPASM (Proteus)
;=====

;=====
; DEFINITIONS
;=====

#include p16f84a.inc           ; Include register definition file

portA    equ 0x05      ;portA address
portB    equ 0x06      ;portB address
trisA    equ 0x05      ;trisA address in register bank 1
trisB    equ 0x06      ;trisB address in register bank 1

status   equ 0x03      ;status register
RP0      equ 5         ;bit 5 of status register
RP1      equ 6         ;bit 6 of status register

;=====
; VARIABLES
;=====

count    equ 0x0c      ;counter 0 to 255
nCount   equ 0x0d      ;pause routine inner loop counter
  
```

```

mCount equ 0x0e    ;pauseroutine outer loop counter

;=====
; RESET and INTERRUPT VECTORS
;=====

    ; Reset Vector
RST   code 0x00
      goto Start

;=====
; CODE SEGMENT
;=====

PGM   code
Start
      bcf   status,RP0    ; clear selection bits in the status reg
      bcf   status,RP1

      bsf   status,RP0    ; reselect bank 0
      movlw 0x00          ; load w with 0
      movwf trisB         ; load it into trisB to make portB all outputs

      movlw 0x03          ; load w with 0x03
      movwf trisA         ; load it into trisA to make the two lsbs inputs

      bcf   status,RP0    ; reselect bank 0

      clrf  portB         ; clear all lines low
      comf  portB,1       ; complement the port to switch all LEDs off
      clrf  count         ; initialize counter to 0

buttons
      btfsc portA,0       ; look to see if left button is pressed
      goto leftbutton

      btfsc portA,1       ; look to see if right button is pressed
      goto rightbutton

leftbutton
      movlw 0x67          ; show 'l' on lcd
      call pause          ;delay for 2 secs
      call pause
      call pause
      call pause
      call pause
      call pause
      call pause
      call pause
      call pause
      call pause
      goto buttons        ; if not pressed, just keep checking

rightbutton
      movlw 0x2F          ; show 'r' on lcd
      call pause          ;delay for 2 secs
      call pause
      call pause
      call pause
      call pause

```

```

call pause
call pause
call pause
call pause
call pause
goto buttons      ; just keep looping

;get_count
; movf  count,w      ; move count to W
; goto characters
; movwf portB        ; move W to portB to update the display
; addlw 1            ; add +1 to W
; movwf count        ; save W in count

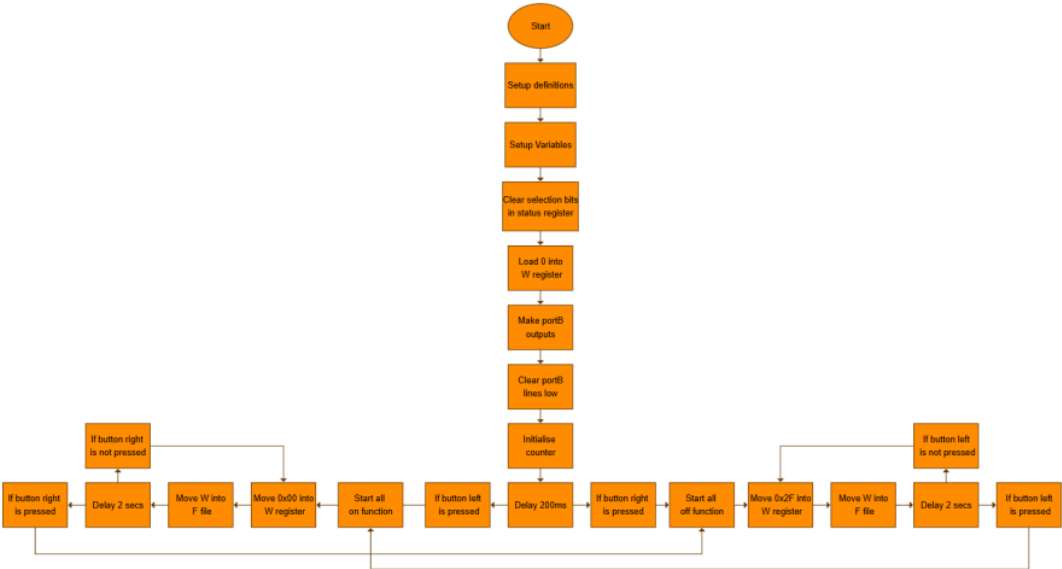
;=====

pause
    movlw 0xff        ; set W = 255 decimal
    movwf mCount      ; mCount = w: outer loop counter
loadN
    movlw 0xff        ; set w = 255 decimal
    movwf nCount      ; nCount = w: inner loop counter
decN
    decfsz nCount, f   ; nCount--
    goto decN          ; if nCount != 0 then repeat nCount--

    decfsz mCount, f   ; else decrement Count
    goto loadN         ; if mCount != 0 then
                        ; reload nCount to 255 and decrement
    return            ; else exit subroutine
;=====
END

```

TASK 3:



References

- [1] "7-segment Display and Driving a 7-segment Display," Basic Electronics Tutorials, 16 February 2018. [Online]. Available: <https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html>. [Accessed 8 January 2025].
- [2] "7805 Voltage Regulator 5V / 1A," ProtoSupplies, 23 November 2024. [Online]. Available: <https://protosupplies.com/product/voltage-regulator-7805-5v-1a/?form=MG0AV3>. [Accessed 8 January 2025].
- [3] "Quartz Crystal Oscillator and Quartz Crystals," Basic Electronics Tutorials, 25 August 2013. [Online]. Available: <https://www.electronics-tutorials.ws/oscillator/crystal.html>. [Accessed 8 January 2025].
- [4] "What Is A Resistor And What Does It Do?," Build Electronic Circuits, 3 October 2013. [Online]. Available: <https://www.build-electronic-circuits.com/what-is-a-resistor/>. [Accessed 8 January 2025].
- [5] "PIC16F84A Microcontroller introduction and features," Microcontrollers Lab, 10 December 2018. [Online]. Available: <https://microcontrollerslab.com/pic16f84a-microcontroller-introduction-and-features/>. [Accessed 8 January 2025].
- [6] "How Do Piezo Buzzers Work," uk.rs-online.com, [Online]. Available: <https://uk.rs-online.com/web/content/discovery/ideas-and-advice/how-do-piezo-buzzers-work>. [Accessed 8 January 2025].
- [7] "Capacitor Types and How Capacitors Are Constructed," Electrical4U, 15 February 2014. [Online]. Available: <https://www.electrical4u.com/what-is-capacitor/>. [Accessed 8 January 2025].
- [8] "NPN Transistor Tutorial - The Bipolar NPN Transistor," Basic Electronics Tutorials, 5 February 2019. [Online]. Available: https://www.electronics-tutorials.ws/transistor/tran_2.html. [Accessed 8 January 2025].
- [9] "Poles and Throws: SPST, SPDT, DPST, DPDT Switches," Langir Electric, 21 April 2024. [Online]. Available: <https://www.langir.com/news/poles-and-throws-spst-spdt-dpst-dpdt-switches/>. [Accessed 8 January 2025].
- [10] "SPST Momentary Push Button," Components 101, 25 August 2018. [Online]. Available: <https://components101.com/switches/spst-momentary-push-button?form=MG0AV3>. [Accessed 8 January 2025].

SB Micros Assignment

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

60/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

GRADING FORM: GJ_2019

SARAH BUTLER**301****ENGAGEMENT WITH LITERATURE**

A good number of relevant sources cited.

73**KNOWLEDGE AND UNDERSTANDING**

While the depth isn't great, the overall K&U of the component operation is acceptable, with the diagrams and images being excellent in helping the explanations

68**COGNITIVE AND INTELLECTUAL**

As this section is for flair and innovation in solving the programming, this section is weak.

40**PRACTICAL APPLICATION**

Circuit is well simulated, but only half the tasks attempted.

50

TRANSFERABLE SKILLS



Generally well written and presented.
The TOC could have been more detailed though.

70