Reporte Practica 6 - PWM

Diseño en sistemas de chip

Diego Armando Limón de León A01638247

Jose Miguel Figarola Prado A01632557

Giancarlo Franco Carrillo A01638108

Tecnológico de Monterrey

xx de mayo del 2021

In this lab the idea is to integrate aspects of the various modules we have seen so far: LCD display, 4x4 matrix keyboard, interrupts, ADC and PWM. Before proceeding, be sure that your code for the LCD and the keyboard works properly. You can use either the 8- or 4-bit option for the LCD code.

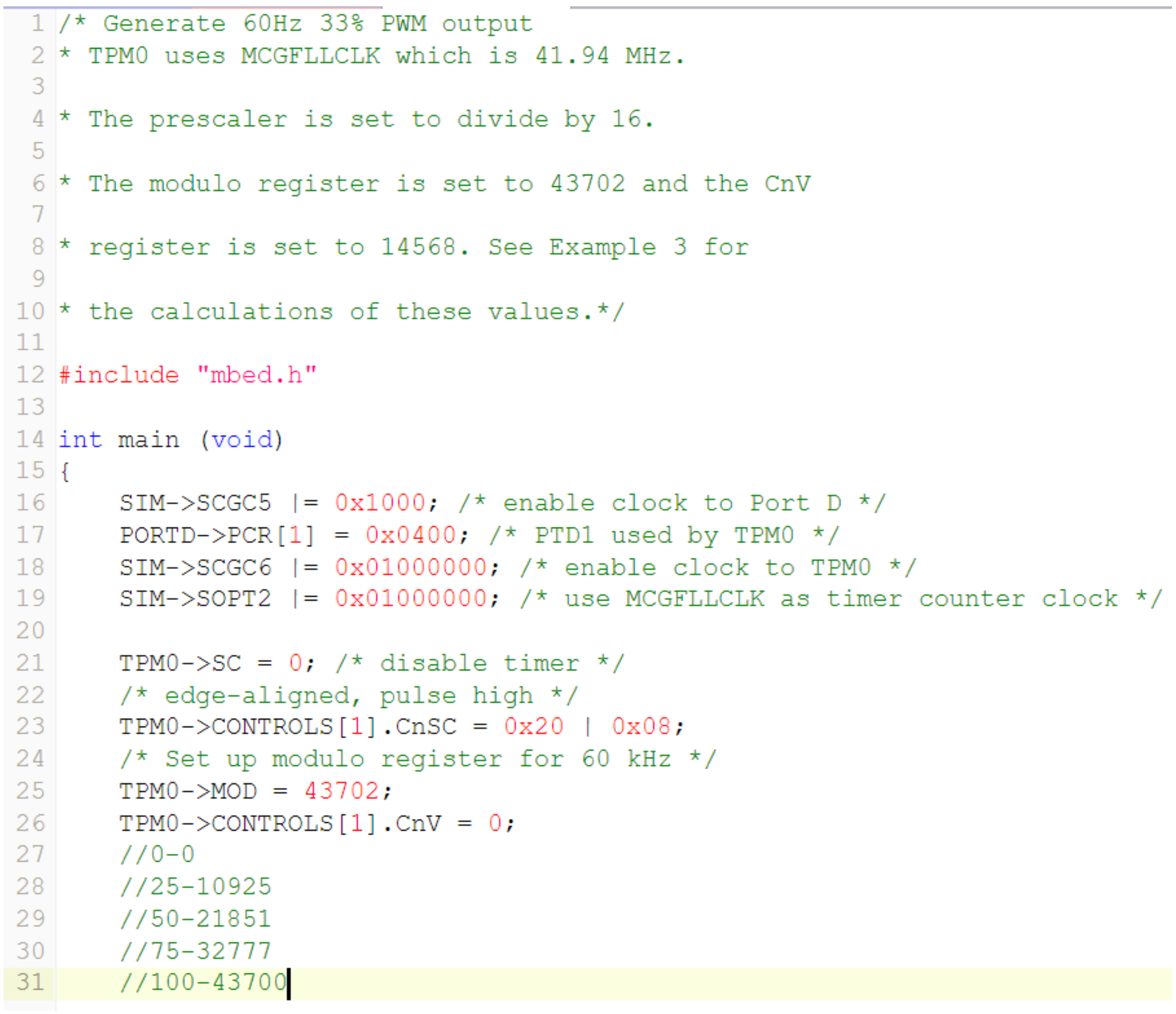
The lab will be divided into four parts, described as follows:

Part 1. PWM Part 1. Modify the first example seen in class to generate PWM signals with a frequency of 60Hz (TPMx_MOD = 43702) but different duty cycles: 0, 25%, 50%,75% and 100% (since we are using non -inverted PWM we will have the LED completely on for 0 DT and off for 100%.

You have to calculate the values for the TPMx_CnV register for the different duty cycle values

Analisis: In this code it was necessary to change the TPMx_CnV in order to change the PWM values:

- TPMx_CnV = 0 -> 100%
- TPMx_CnV = 10925 -> 75%
- TPMx_CnV = 21851 ->50%
- TPMx_CnV = 32777 -> 25%
- TPMx_CnV = 43703 -> 0%

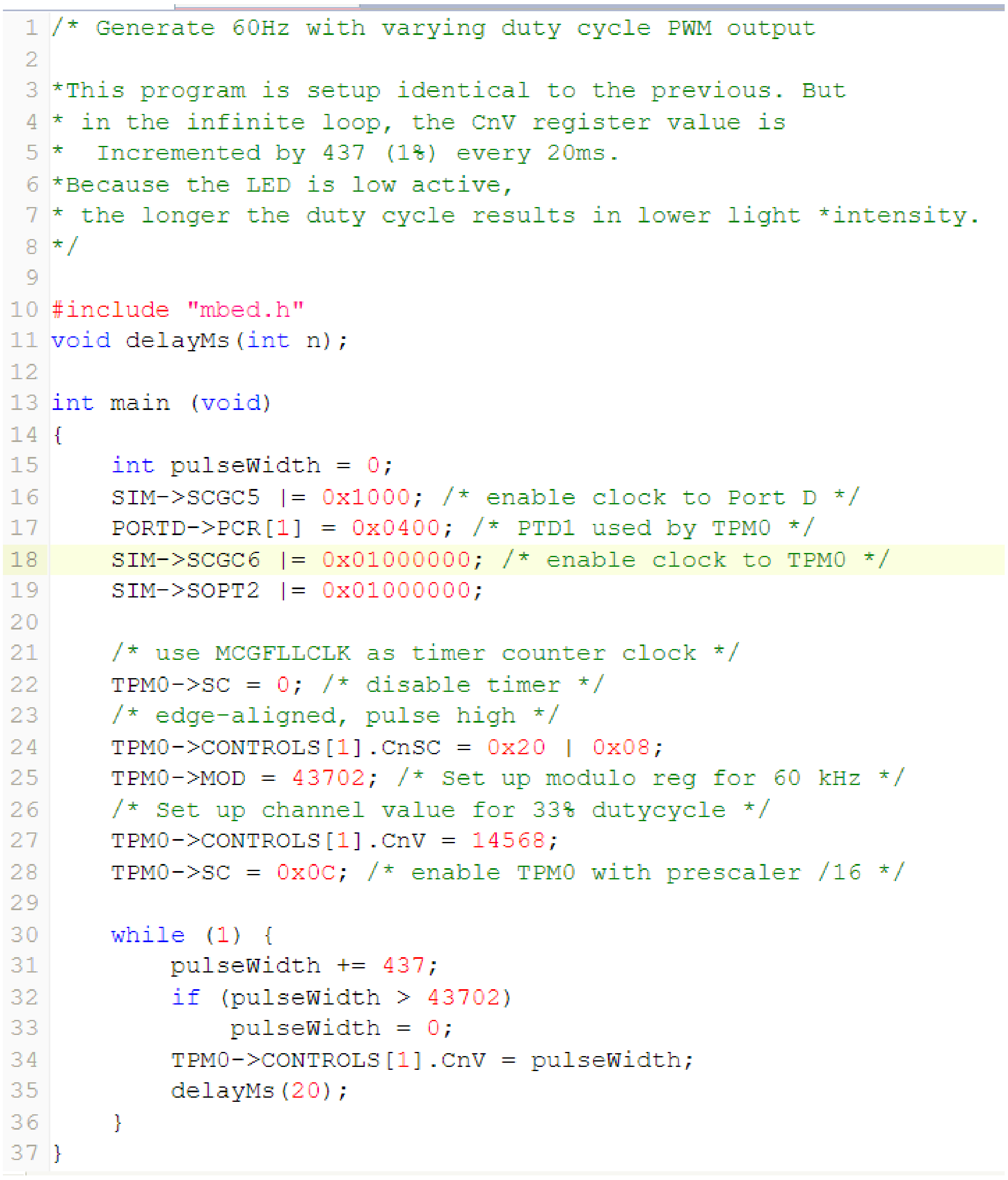


```
 1 /* Generate 60Hz 33% PWM output
 2 * TPM0 uses MCGFLLCLK which is 41.94 MHz.
 3
 4 * The prescaler is set to divide by 16.
 5
 6 * The modulo register is set to 43702 and the CnV
 7
 8 * register is set to 14568. See Example 3 for
 9
10 * the calculations of these values.*/
11
12 #include "mbed.h"
13
14 int main (void)
15 {
16     SIM->SCGC5 |= 0x1000; /* enable clock to Port D */
17     PORTD->PCR[1] = 0x0400; /* PTD1 used by TPM0 */
18     SIM->SCGC6 |= 0x01000000; /* enable clock to TPM0 */
19     SIM->SOPT2 |= 0x01000000; /* use MCGFLLCLK as timer counter clock */
20
21     TPM0->SC = 0; /* disable timer */
22     /* edge-aligned, pulse high */
23     TPM0->CONTROLS[1].CnSC = 0x20 | 0x08;
24     /* Set up modulo register for 60 kHz */
25     TPM0->MOD = 43702;
26     TPM0->CONTROLS[1].CnV = 0;
27     //0-0
28     //25-10925
29     //50-21851
30     //75-32777
31     //100-43700
```

As we can see from the description comments from the code,we use the 60Hz and the TMP0 module which runs at 41.94 Mhz. The modulo register is set 43702. By changing the line 26 you can change the PWM percentage, you can see we are using 0 as TMPx_CnV so it returns 100% of the PWM output.

Part 2. PWM Part 2. Implement the code for the second example seen in class, in which the PWM signal is changed by increments of 1% in the CnV register. Observe how the intensity of the LED decreases as we increment the duty cycle (because the LED is active low)

Analisis: In this part, the PWM percentage changes by 1% every 1 second and the LED intensity changes from 100% to 0%. In the code, the CnV register changes with an increment of 437 every 20ms. As shown in line 30 we use a while loop that changes the pulse width that initially started in 0 by adding 437 and if the PW is lower than 43702 it restarts the PW.



```c
/* Generate 60Hz with varying duty cycle PWM output

*This program is setup identical to the previous. But
* in the infinite loop, the CnV register value is
*  Incremented by 437 (1%) every 20ms.
*Because the LED is low active,
* the longer the duty cycle results in lower light *intensity.
*/

#include "mbed.h"
void delayMs(int n);

int main (void)
{
    int pulseWidth = 0;
    SIM->SCGC5 |= 0x1000; /* enable clock to Port D */
    PORTD->PCR[1] = 0x0400; /* PTD1 used by TPM0 */
    SIM->SCGC6 |= 0x01000000; /* enable clock to TPM0 */
    SIM->SOPT2 |= 0x01000000;

    /* use MCGFLLCLK as timer counter clock */
    TPM0->SC = 0; /* disable timer */
    /* edge-aligned, pulse high */
    TPM0->CONTROLS[1].CnSC = 0x20 | 0x08;
    TPM0->MOD = 43702; /* Set up modulo reg for 60 kHz */
    /* Set up channel value for 33% dutycycle */
    TPM0->CONTROLS[1].CnV = 14568;
    TPM0->SC = 0x0C; /* enable TPM0 with prescaler /16 */

    while (1) {
        pulseWidth += 437;
        if (pulseWidth > 43702)
            pulseWidth = 0;
        TPM0->CONTROLS[1].CnV = pulseWidth;
        delayMs(20);
    }
}
```

Part 3. Simple industrial control with PWM. Imagine that we want to create an industrial grinder. Usually, just like a with house blender, these systems have several power configurations depending on how “hard” are the components (i.e. gravel, stones) we wish to grind and thus the motor changes speed (and thus torque) to do the work. In this sense, we will create a simple application that can change the speed depending on a setting defined by the user

1. When you start your application, the next message should be displayed.

Set input mode

Mode 1: M Mode 2: A

Mode M stands for manual and mode 2 stands for Automatic. We will see what each mode implies. You can use each mode by pressing whichever button in the keyboard you choose and pressing # toe execute the rest of your application

2. In manual mode, the idea is that you integrate the first part of this lab, but modifying the code for generating inverted PWM signals. Thus, the LCD should display

Select Speed

1: L 2:M 3:MH 4:H

(standing for Low, Medium, Medium High and High)

After pressing the one of the keys the corresponding CnV register values should be sent to a function the initializes the PWM (duty cycles of 25%, 50%, 75% and 100%). Please send the value as an argument to the function (if you use if statements the code becomes very long and it is very inefficient). The LED in this case will go from very low intensity to completely on (as the case of a motor). If you have a fan or a small motor you can try as we do not require and H bridge

2. After this, if we want to modify the speed of the motor, we can use a push button to send an interrupt to the MCU and display the above menu again. A second button can be used to stop the system the motor at any given time (emergency signal)

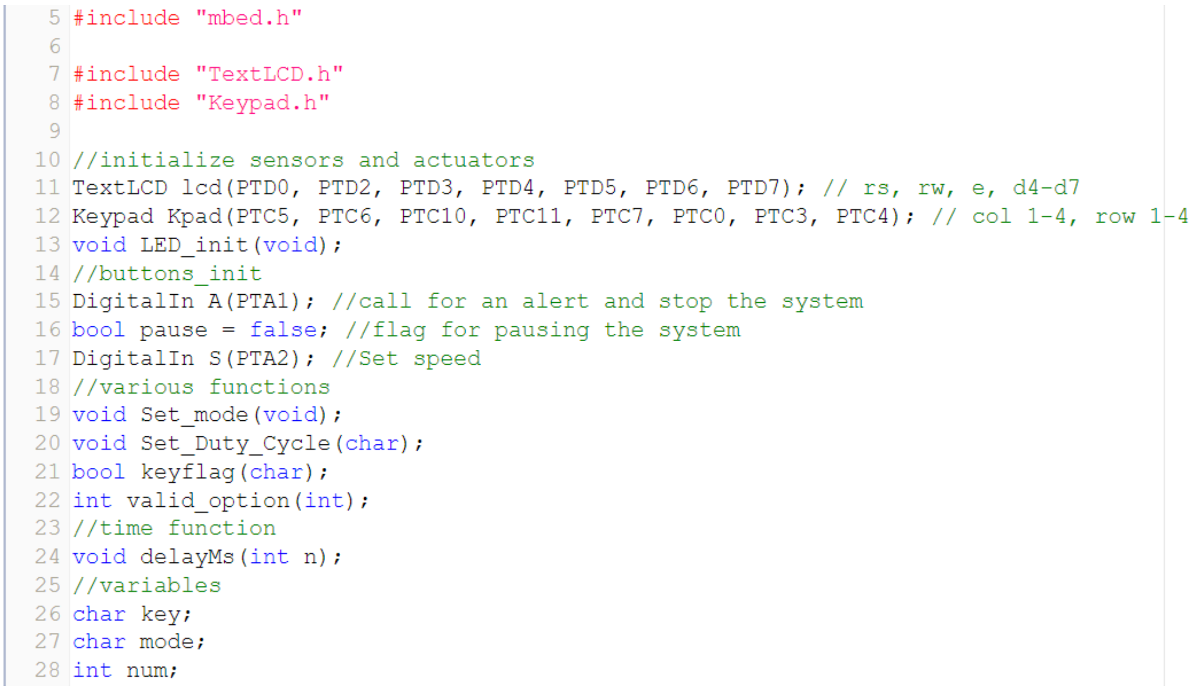
Just as a note, the ADC could be used here to monitor the motor current or heat and send a signal to the microcontroller if there is an overload (the system cannot grind something for instance) and increase the power or switch the motor off. Something in the context is explored in the next part of the lab (optional)

Automatic Mode (extra points)

1. This system is very similar to the previous case: there are 4 predefined power levels, which are chosen depending on the value of the CnV register. However, in this case, we continuously monitor the value of the ADC connected to a potentiometer (simulating the load of the motor). If the value is between 0 – 0.75V the motor should run on Mode 1; between 0.76 and 1.5V on Mode 2; between 1.51 and 2.25V on mode 3 and finally, between 2.25 and 3V on Mode 4. We also should use a button to go to the main menu and a second button to stop the motor.

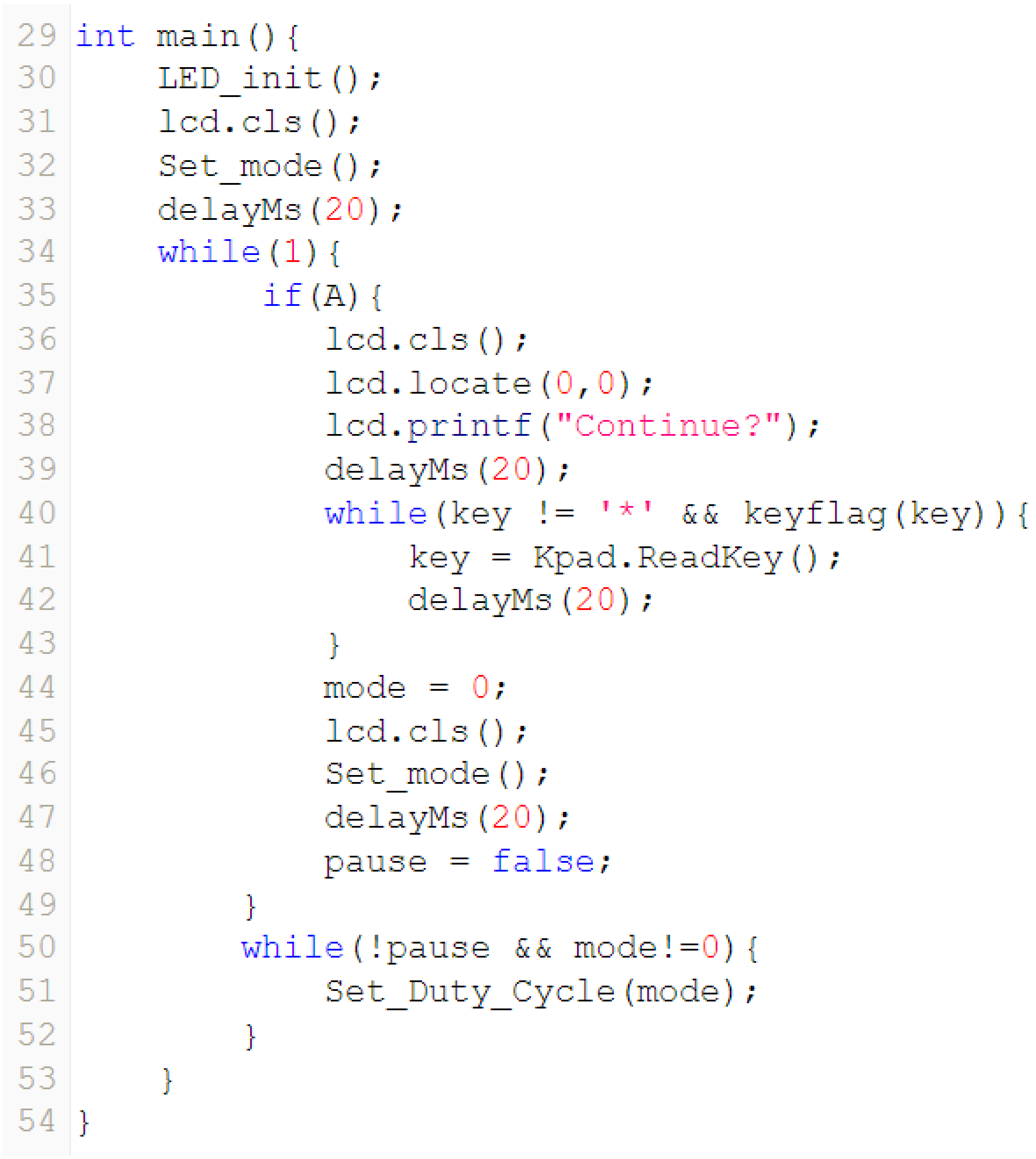
Analisis: For this part we use PWM knowledge from the other parts in order to create a menu that displays 2 options: Manual or Automatic. In the manual option you can choose the LED's brightness with 4 different options that go from 0 to 100 percent with a step size of 25 (L,M,MH,H). The automatic option increases the PWM input and the LED´s brightness changes every certain second. The code implements an interrupt with a pushbutton to pause the system.

First of all we declare the objects as the LCD and the Keypad with their respective libraries. The interrupt input is declared in line 15 and we are using a flag to clean or activate the interrupt.



```
 5 #include "mbed.h"
 6 
 7 #include "TextLCD.h"
 8 #include "Keypad.h"
 9 
10 //initialize sensors and actuators
11 TextLCD lcd(PTD0, PTD2, PTD3, PTD4, PTD5, PTD6, PTD7); // rs, rw, e, d4-d7
12 Keypad Kpad(PTC5, PTC6, PTC10, PTC11, PTC7, PTC0, PTC3, PTC4); // col 1-4, row 1-4
13 void LED_init(void);
14 //buttons_init
15 DigitalIn A(PTA1); //call for an alert and stop the system
16 bool pause = false; //flag for pausing the system
17 DigitalIn S(PTA2); //Set speed
18 //various functions
19 void Set_mode(void);
20 void Set_Duty_Cycle(char);
21 bool keyflag(char);
22 int valid_option(int);
23 //time function
24 void delayMs(int n);
25 //variables
26 char key;
27 char mode;
28 int num;
```

The main function of the program initializes the LCD and calls the function Set_mode() which lets the user select the automatic or manual mode. If the pushbutton is pressed, a message is sent to the LCD and waiting to press the # button to continue the program.

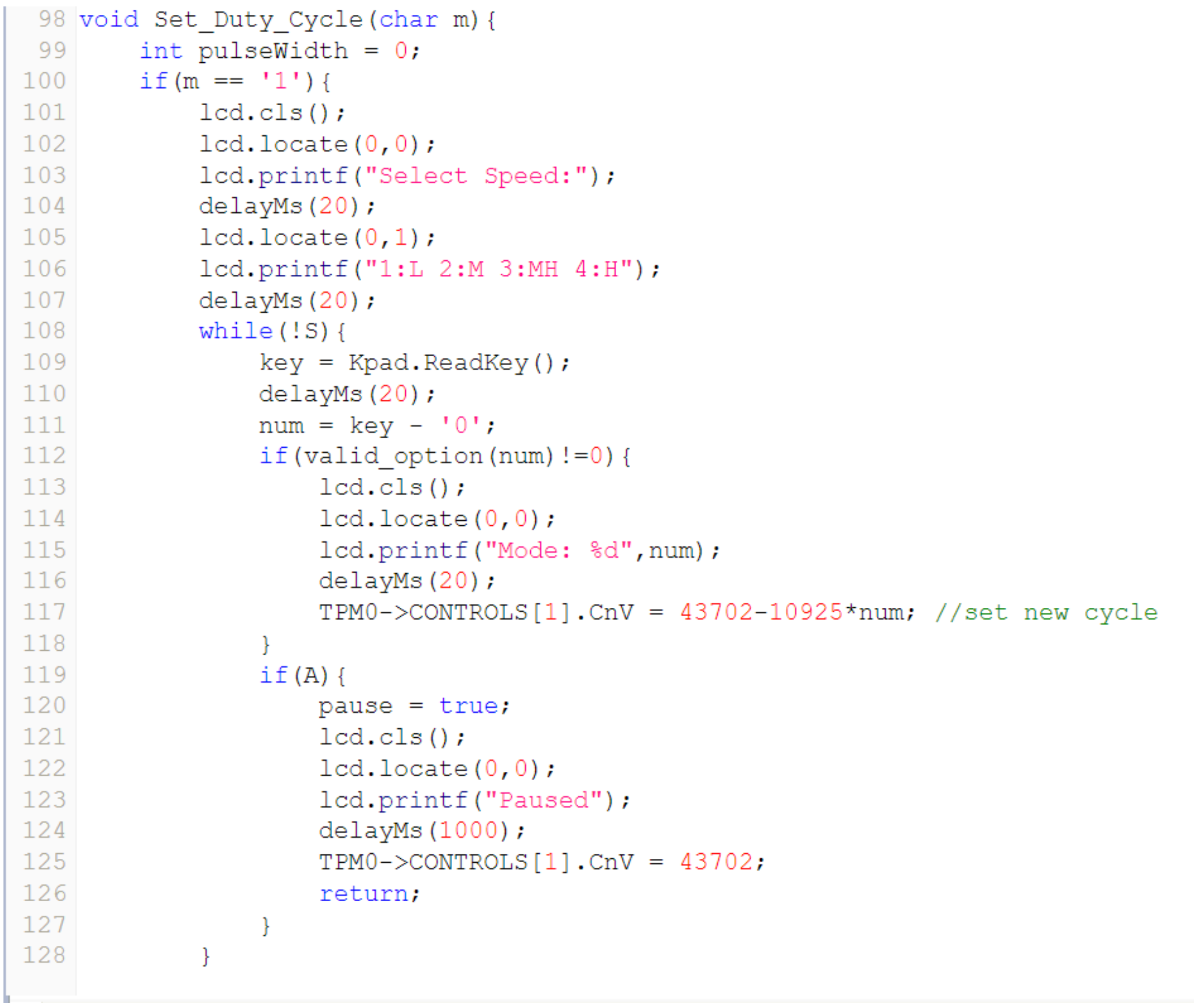


```
int main(){
    LED_init();
    lcd.cls();
    Set_mode();
    delayMs(20);
    while(1){
         if(A){
            lcd.cls();
            lcd.locate(0,0);
            lcd.printf("Continue?");
            delayMs(20);
            while(key != '*' && keyflag(key)){
                key = Kpad.ReadKey();
                delayMs(20);
            }
            mode = 0;
            lcd.cls();
            Set_mode();
            delayMs(20);
            pause = false;
        }
        while(!pause && mode!=0){
            Set_Duty_Cycle(mode);
        }
    }
}
```

The Set_mode() function lets the user decide within both of the options, and the displayed messages shows up until the # key is pressed after selecting a mode. When the manual mode is selected the LCD displays an “M” and when the automatic mode is pressed it displays an “A”.

```
73 void Set_mode(void){
74     lcd.cls();
75     lcd.locate(0,0);
76     lcd.printf("Set Mode:");
77     delayMs(20);
78     lcd.locate(0,1);
79     lcd.printf("1:M 2:A");
80     delayMs(20);
81     while(key != '#' && keyflag(key)){//set mode
82     key = Kpad.ReadKey();
83     delayMs(20);
84     num = key - '0';
85         if(key == '1'){
86             lcd.locate(9,0);
87             lcd.printf("M"); //display the mode
88             mode = '1';
89         }
90          if(key == '2'){
91             lcd.locate(9,0);
92             lcd.printf("A"); //display the mode
93             mode = '2';
94         }
95     }
96 }
```

The Set_Duty_Cycle has two parts, when the Manual mode is selected, it goes through the first if statement where another message is displayed on the LCD, this is where the user can select within the four different manual modes. Depending on the mode selected the CnV value changes as shown in line 117. When the pushbutton is pressed while running the code, it enters to a pause mode and the LED turns off.



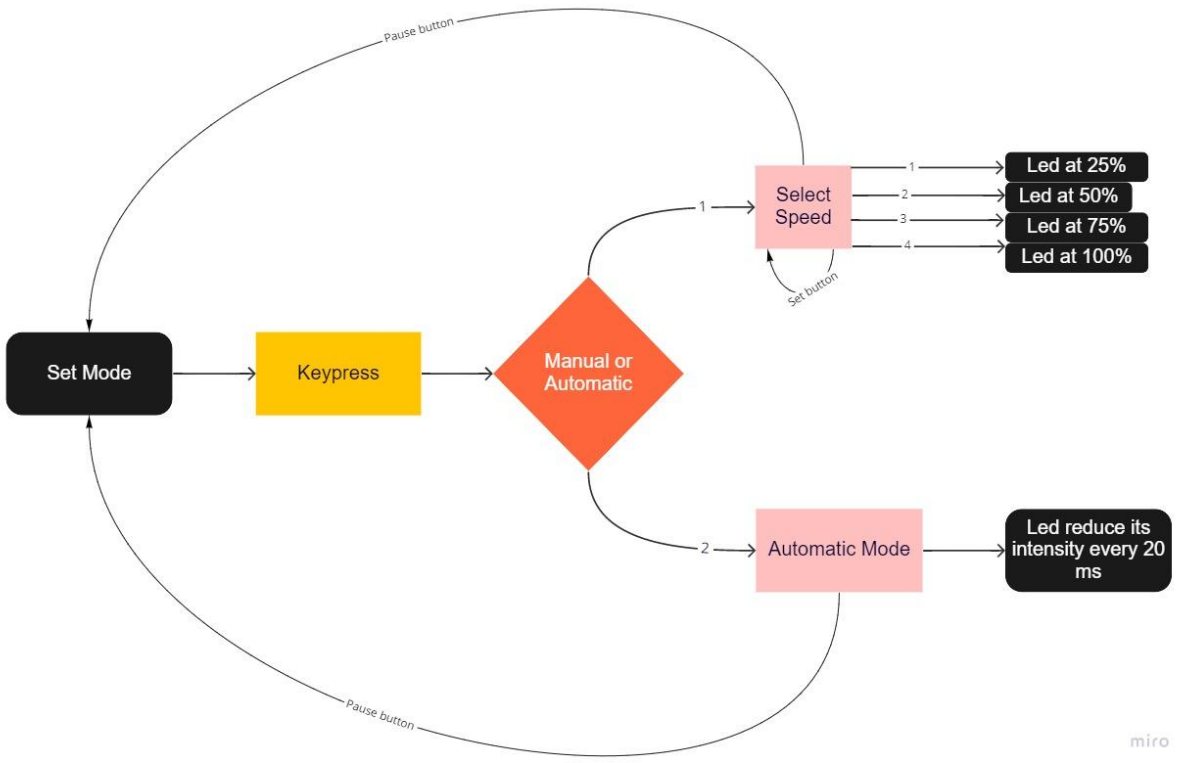
```
void Set_Duty_Cycle(char m){
    int pulseWidth = 0;
    if(m == '1'){
        lcd.cls();
        lcd.locate(0,0);
        lcd.printf("Select Speed:");
        delayMs(20);
        lcd.locate(0,1);
        lcd.printf("1:L 2:M 3:MH 4:H");
        delayMs(20);
        while(!S){
            key = Kpad.ReadKey();
            delayMs(20);
            num = key - '0';
            if(valid_option(num)!=0){
                lcd.cls();
                lcd.locate(0,0);
                lcd.printf("Mode: %d",num);
                delayMs(20);
                TPM0->CONTROLS[1].CnV = 43702-10925*num; //set new cycle
            }
            if(A){
                pause = true;
                lcd.cls();
                lcd.locate(0,0);
                lcd.printf("Paused");
                delayMs(1000);
                TPM0->CONTROLS[1].CnV = 43702;
                return;
            }
        }
```

The second part of the function is when the automatic mode is chosen. As well, if the push button is pressed it enters a pause mode.The pulsewidth has a change of 437 every second so the user is not responsible of the brightness.

```
    if(m == '2'){
        while(!S) {
            lcd.cls();
            lcd.locate(0,0);
            lcd.printf("Automatic Mode");
            delayMs(20);
            if(A){
                pause = true;
                lcd.cls();
                lcd.locate(0,0);
                lcd.printf("Paused");
                delayMs(1000);
                TPM0->CONTROLS[1].CnV = 43702;
                return;
            }
            pulseWidth += 437;
            if (pulseWidth > 43702)
                pulseWidth = 0;
            TPM0->CONTROLS[1].CnV = pulseWidth;
            delayMs(20);
        }
        return;
    }
    return;
}
```

Video

For a better visualization of the codes, a YouTube video was made in order to show the results: https://youtu.be/GvkYvV5msh4


Pause button
Select Speed
1
2
3
4
Led at 25%
Led at 50%
Led at 75%
Led at 100%
Set button
1
Set Mode
Keypress
Manual or Automatic
2
Automatic Mode
Led reduce its intensity every 20 ms
Pause button
miro