Wireless Motor PWM

Includes:

I. Problem Description
II. Pseudocode
III. Assembly Code (with comments)
IV. Wiring Diagram
V. Video of Systems Functionality

By: Brandt Ousley

View Video Here:

h"ps://youtu.be/M-Fn3gKDCe8

I. Problem Description

The objective of the **Final Project: Wireless Motor PWM** is to design a system using a ESP32 Wi-Fi chip with the use of a serial terminal app to control a PWM output signal for a 12 VDC electric motor, displaying the speed percentage (0 to 100) on a 2x16 I2C-connected LCD.

My systems functionality: I used a 9v battery pack, DC motor, 2x16 LCD screen, MSP430, ESP32 Wi-Fi chip, TCP Wi-Fi app on iOS, a breadboard, jumper wires, and a blue fan connected to the motor.

II. Pseudocode

--- main.c Pseudocode ----

Initialize the necessary libraries and headers for the MSP430 microcontroller.

In the main function:
- Stop the watchdog timer to prevent it from resetting the microcontroller.
- Set up the timers, I2C, and Wi-Fi by calling their initialization functions.
- Initialize the I2C communication with the slave address 0x27.
- Configure pin 2.0 for PWM (Pulse Width Modulation) output.
- Configure pin 2.1 to control the motor direction.
- Enable global interrupts and turn off high impedance mode.

- Set up the LCD screen and display the text "Motor:".

Enter an infinite loop:
- Wait for the client to connect and send input.
  - During this waiting period, the timer will start based on the received data, and a flag will be set upon overflow.

- Introduce a delay to wait for client input.

- If the entered value is less than 100:
  - Enable the CCR1 interrupt.
  - Set the PWM duty cycle based on the entered value.
  - Clear the LCD display.
  - Write the text "Motor: " followed by the entered value and a percentage sign on the LCD.

- Otherwise:
  - Disable the CCR1 interrupt.
  - Clear the LCD display.
  - Write the text "Motor: 100%" on the LCD.

- Disable the TB1 interrupt and clear the interrupt flag.
- Reset the entered value and the completion flag for the next input.

--- WIFI.C Pseudocode ----

Include necessary libraries and headers for the MSP430 microcontroller and Wi-Fi.

Define global variables:
- stores the value entered by the user.
- flag to indicate if input is complete.

Define the function to initialize Wi-Fi:
- Enable software reset.
- Select SMCLK (sub-main clock) as the clock source.
- Set the baud rate to 9600 bps.
- Configure modulation for the UART.
- Set pin P1.6 to UART mode.
- Disable software reset to take the UART out of reset mode.
- Enable the RX (receive) interrupt.

Define the interrupt service routine (ISR) for UART receive:
- Read the received value from the UART buffer.
- If the received value is a digit ('0' to '9'):
  - Update by multiplying the current value by 10 and adding the new digit.
  - Enable TB1 interrupts to start the timer.
- Restart the timer by setting its register to 1.

III. Assembly Code (with comments) Main, Wifi.c/h, ESP32

```
#include <msp430.h>
#include "LiquidCrystal_I2C.h"
#include "timer.h"
#include "wifi.h"
#include <stdio.h>

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;     // stop watch dog timer

// call timer, ic2, and wifi functions
    timer_b0_setup();
    timer_b1_setup();
    wifi_init();
    I2C_Init(0x27); // slave address

// set 2.0 for pwm output
    P2DIR |= BIT0;
    P2OUT &= ~BIT0;

// setting the motor direction, 2.1
        P2DIR |= BIT1;
        P2OUT |= BIT1;

// global enable/turn off high impedance
    PM5CTL0 &= ~LOCKLPM5;
    __enable_interrupt();
// setting up the LCD screen, the cursor and
// and writing string 'motor'
        LCD_Setup();
        LCD_SetCursor(0, 0);
        LCD_Write("Motor:");

    while (1)
    {
```

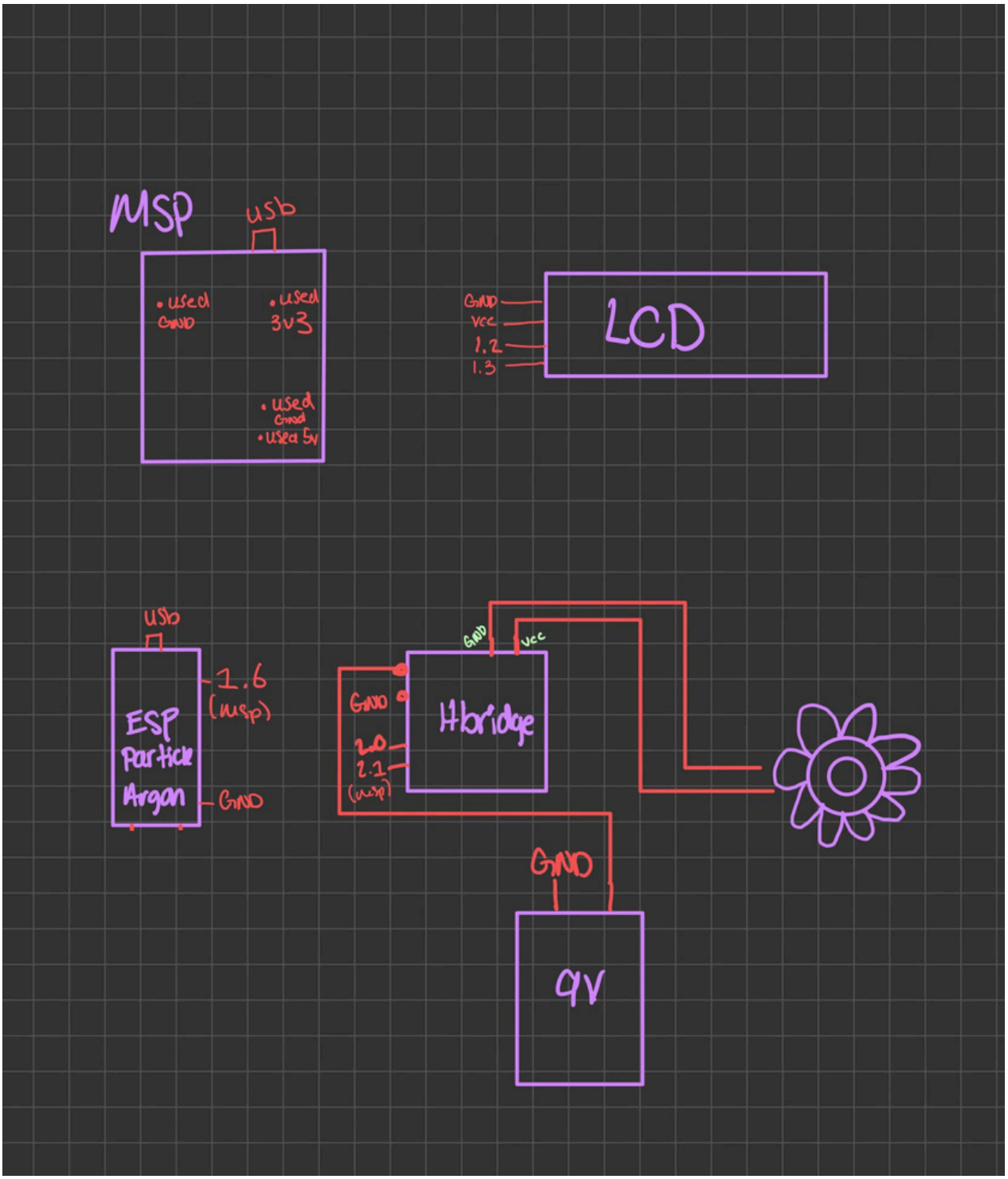
```
 while (1)
 {
// waiting for client connected to send over input
     while(if_complete == 0)
     {

         // Here were waiting for data to be sent over which tells
         // timer b to start and set is complete to 1 upon overlfow
     }
     int i;
     for(i=0;i<10000;i++){}      // delay to wait for client input

     // if less then 100, no overflow occurs
     if (entered_value < 100)
     {
         TB0CCTL1 |= CCIE;                 // CCR1 interrupt
         TB0CCR1 = (entered_value * PERIOD) / 100;
         LCD_ClearDisplay();             // clear the display
         LCD_Write("Motor: ");          // writing constant string
         LCD_WriteNum(entered_value);  // value that is sent over
         LCD_SetCursor(9,0);          // setting cursor length to fit
         LCD_Write("%");
     }
     // when if is false
     else
     {
         TB0CCTL1 &= ~CCIE;           // enable CCR1 interrupt
         LCD_ClearDisplay();         // clear display
         LCD_Write("Motor: 100%");  // set value to 100% as max
     }
     TB1CCTL0 &= ~CCIE;      // disable TB1 -> interrupt
     TB1CCTL0 &= ~CCIFG;    // lower flag here
     entered_value = 0;    // reset for the new value
     if_complete = 0;     // rst for another input
 }
```

```c
#include <msp430.h>
#include "wifi.h"

unsigned int entered_value = 0;
unsigned int if_complete = 0;

void wifi_init(void) {
    UCA0CTLW0 |= UCSWRST;           // Software reset
    UCA0CTLW0 |= UCSSEL__SMCLK;     // Select SMCLK
    UCA0BRW = 104;                  // Baud rate setting for 9600 bps
    UCA0MCTLW = UCBRS0;             // Modulation
    P1SEL0 |= BIT6;                 // Set P1.6 to UART mode
    P1SEL1 &= ~BIT6;

    UCA0CTLW0 &= ~UCSWRST;          // Take out of software reset
    UCA0IE |= UCRXIE;               // Enable RX interrupt
}

#pragma vector = EUSCI_A0_VECTOR
__interrupt void ISR_A0_RX(void) {
    unsigned int value = UCA0RXBUF;
    if (value >= '0' && value <= '9') {
        entered_value = (entered_value * 10) + (value - '0');
        TB1CCTL0 |= CCIE; // Enable TB1 interrupts... start timer
    }
    TB1R = 1; // Restart timer
}
```

```c
/*

 * For comprehensive documentation and examples, please visit:
 * https://docs.particle.io/firmware/best-practices/firmware-template/
 */

#include "Particle.h"

// Run the application and system concurrently in separate threads
// Show system, cloud connectivity, and application logs over USB
SerialLogHandler logHandler(LOG_LEVEL_INFO);

// Ensure the system is in manual
SYSTEM_MODE(MANUAL);
SYSTEM_THREAD(ENABLED);




// TCP Server setup
TCPServer server = TCPServer(8080); // Port number
TCPClient client;

// Global variable to store the IP address
```

```
String ipAddress;

void setup() {
    Serial.begin(9600); // Start serial communication
    Serial1.begin(9600); // Start Serial1 for communication with MSP430
    server.begin(); // Start the TCP server

    // Ensure Wi-Fi is connected
    WiFi.on();
    WiFi.connect();
    while (!WiFi.ready()) {
        Particle.process(); // Keep system running while waiting for Wi-Fi
        delay(500);
    }

    // Get and print the initial IP address
    ipAddress = WiFi.localIP().toString();
    Serial.print("Device IP Address: ");
    Serial.println(ipAddress);

    // Publish the IP address to Particle Cloud
    Particle.publish("DeviceIP", ipAddress, PRIVATE);
}

void loop() {
    if (client.connected()) {
        while (client.available()) {
            char data = client.read(); // Read data from TCP
            Serial1.write(data); // Send data to MSP430
            Serial.write(data);
        }
    } else {
        client = server.available(); // Accept new client connection
        if (client) {
            Serial.println("New client connected.");
        }
```

IV. Wiring Diagram


MSP
usb
• used GND
• used 3v3
• used Gnad
• used 5v
GND
Vcc
1.2
1.3
LCD
usb
ESP Particle Argon
1.6 (msp)
GND
GND
Vcc
GND
2.0
2.1 (msp)
Hbridge
GND
9V

V. Video of Circuit - YouTube Link

h"ps://youtu.be/M-Fn3gKDCe8