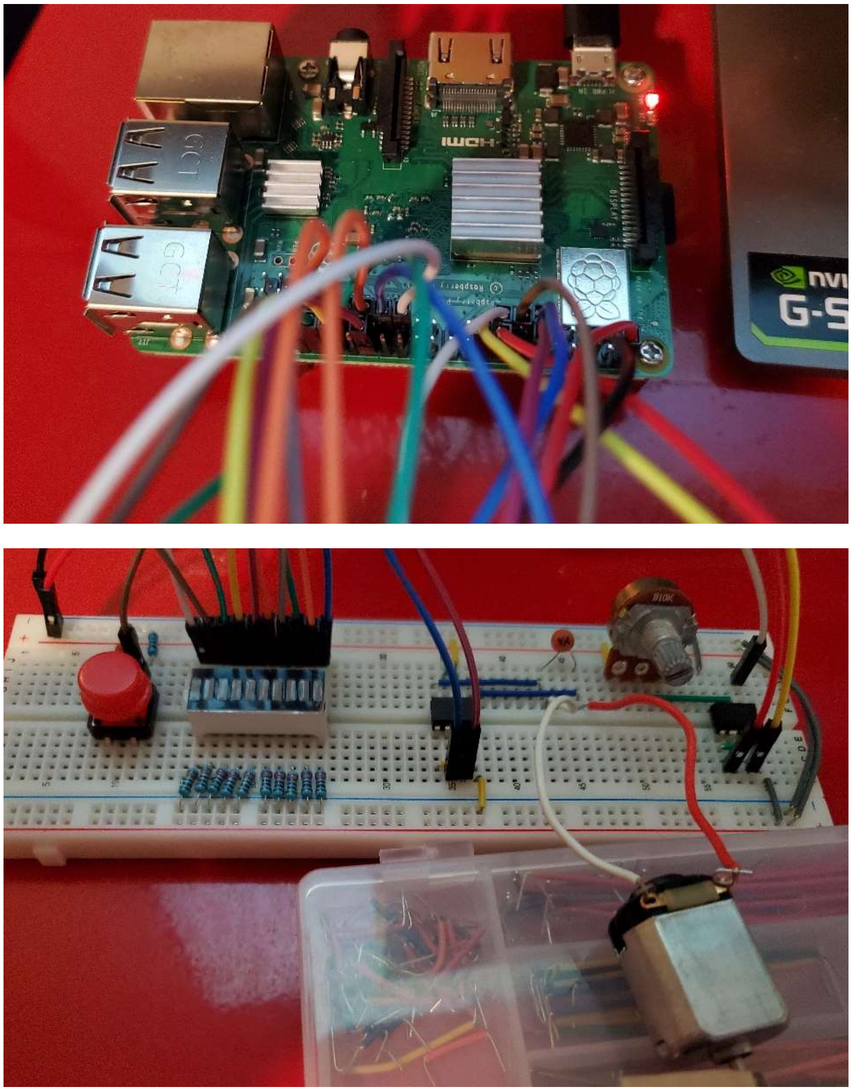
Jordan Ditzler

G00967092

Lab 5 Report

November 1, 2020

Photo of RPi & Board



The button is connected to power via 1 kΩ resistor and has a 0.1 µF capacitor in series with it for help with debouncing. The debouncer circuit is not perfect, but paired with software debouncing, it eliminates most transients.

Each LED of the 10-LED bar is in series with 1 220 Ω resistor to limit current.

The L9110 H-bridge chip is set up exactly like lesson #22 in the Adeept manual w/ a 0.1 µF capacitor in parallel with the DC motor and the IA & IB pins connected to RPi GPIO.

The 10 kΩ potentiometer is set up exactly like lesson #19 in the Adeept manual w/ the left pin connected to Vcc/ the middle pin connected to the CH0 pin on the ADC0832, and the right pin connected to GND.

The ADC setup is also from lesson #19 w/ DI & DO sharing a GPIO slot and the CS and CLK pins connected to the RPi via GPIO.

NOTE: Since I do not own a soldering kit, and therefore could not solder wires to the DC motor, I must hold it partially in the video submission to make sure its connections are secure and it can run properly.

Code

```c
#include <wiringPi.h>
#include <softPwm.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>

#define IA      0   // Input A pin on L9110
#define IB      1   // Input B pin on L9110

#define BTN     2   // BTN pin

#define ADC_CS      3   // ADC CS pin
#define ADC_DIO     4   // ADC DI/O pin
#define ADC_CLK     5   // ADC CLK pin

#define LED1        21  // LEDs
#define LED2        22
#define LED3        23
#define LED4        24
#define LED5        25
#define LED6        26
#define LED7        27
#define LED8        28
#define LED9        29
#define LED10       30

#define HALFPERIOD  5   // Half period for wait cmds

char dirFLAG, ADC_val, ADC_mod;
int speed;

// Button ISR
void btn_Interrupt(void)
{
    // Delay for debouncing
    delay(50);

    // Changes direction FLAG value between 0 and 1
    if (!digitalRead(BTN)) dirFLAG = (dirFLAG) ? 0 : 1;
    printf("Interrupt! Flag: %d\n", dirFLAG);

    // While wait loop for debouncing
    while(!digitalRead(BTN));
}

// Sets all LEDs that should be on
void ledSet(void)
{
    int i;

    // Sets all LEDs off to clean-up
    for (i = LED1; i < LED10+1; i++) digitalWrite(i, LOW);

    // Sets all LEDs which should be on to be on
    for (i = LED1; i < LED1+ADC_mod; i++) digitalWrite(i, HIGH);
```

```
}

// Simple method to go to the next clock cycle
void next_CLK(void)
{
    // Set CLK to HIGH for one 1/2 period and to LOW for one 1/2 period
    digitalWrite(ADC_CLK, HIGH);
    delayMicroseconds(HALFPERIOD);
    digitalWrite(ADC_CLK, LOW);
    delayMicroseconds(HALFPERIOD);
}

// Reads the current ADC value
void read_ADC(void)
{
    unsigned char data_1 = 0, data_2 = 0;
    int i;

    // Sets up the required start signals to read from the ADC
    // Follows pp. 9, Fig. 19 of the ADC0832-N datasheet
    digitalWrite(ADC_CS, LOW);

    digitalWrite(ADC_CLK,LOW);
    digitalWrite(ADC_DIO,HIGH);
    delayMicroseconds(HALFPERIOD);

    digitalWrite(ADC_CLK,HIGH);
    delayMicroseconds(HALFPERIOD);

    digitalWrite(ADC_CLK,LOW);
    digitalWrite(ADC_DIO,HIGH);
    delayMicroseconds(HALFPERIOD);

    digitalWrite(ADC_CLK,HIGH);
    delayMicroseconds(HALFPERIOD);

    digitalWrite(ADC_CLK,LOW);
    digitalWrite(ADC_DIO,LOW);
    delayMicroseconds(HALFPERIOD);

    digitalWrite(ADC_CLK,HIGH);
    digitalWrite(ADC_DIO,HIGH);
    delayMicroseconds(HALFPERIOD);

    digitalWrite(ADC_CLK,LOW);
    digitalWrite(ADC_DIO,HIGH);
    delayMicroseconds(HALFPERIOD);

    // Sets DIO pin to take input after ADC is ready to be read
    pinMode(ADC_DIO, INPUT);

    // Reads ADC MSB -> LSB
    // Stores value into a 1 byte variable
    // Left shifts old variable and ORs it with current bit value
    for (i = 0; i < 8; i++)
    {
        next_CLK();
        data_1 = data_1 << 1 | digitalRead(ADC_DIO);
    }

    // Reads ADC LSB -> MSB
    // Stores value into a 1 byte variable
    // Left shifts current bit value "i" times and ORs it with old variable
    for (i = 0; i < 8; i++)
    {
        data_2 = data_2 | digitalRead(ADC_DIO) << i;
        next_CLK();
    }

    // Sets ADC back up to take input for next read
    digitalWrite(ADC_CS, HIGH);
    pinMode(ADC_DIO, OUTPUT);

    // Checks for errors by comparing MSB -> LSB and LSB -> MSB values
    ADC_val = (data_1 == data_2) ? data_1 : 0;

    // Prints error message if ADC catches error
    if (!(data_1 == data_2)) printf("ADC error! Data mismatch\n");
}

int main(void) {
    // Sets up WiringPi
    if (wiringPiSetup() < 0)
    {
        printf("Setup wiringPi failed!\n");
        return -1;
    }

    // Sets up BTN ISR
    if (wiringPiISR(BTN, INT_EDGE_FALLING, &btn_Interrupt) < 0)
    {
        fprintf (stderr, "Unable to setup ISR: %s\n", strerror (errno));
        return -1;
    }

    // Sets up motor controller and sets speed at 5 (1/2 speed)
    pinMode(IA, OUTPUT);
    pinMode(IB, OUTPUT);
    softPwmCreate(IB, 0, 10);
    speed = 5;

    // Sets up button
    pinMode(BTN, INPUT);
    pullUpDnControl(BTN, PUD_UP);

    // Sets up LEDs
    int i;
    for (i = LED1; i < LED10+1; i++) pinMode(i, OUTPUT);

    // Sets up ADC
    pinMode(ADC_CS, OUTPUT);
    pinMode(ADC_DIO, OUTPUT);
    pinMode(ADC_CLK, OUTPUT);
    // Sets up ADC initial values
    digitalWrite(ADC_CS, HIGH);
    digitalWrite(ADC_DIO, LOW);
    digitalWrite(ADC_CLK, LOW);

    while(1)
    {
        // Reads ADC value
        read_ADC();
        printf("Analog Value: %d\n", ADC_val);

        // Changes 8-bit value (max 255) to value between 0 and 10
        ADC_mod = ADC_val * 10 / 255;

        // Sets LEDs
        ledSet();

        // Simple speed calculation
        speed = ADC_mod;
        printf("Speed: %d\n", speed);

        if (!dirFLAG)
        {
            // Turns the motor clockwise and sets speed accordingly
            printf("Clockwise\n");
            digitalWrite(IA, HIGH);
            softPwmWrite(IB, 10-speed);
        } else
        {
            // Turns the motor anti-clockwise and sets speed accordingly
            printf("Anti-Clockwise\n");
            digitalWrite(IA, LOW);
            softPwmWrite(IB, speed);
        }

        delay(500);
    }
}
```