Victor Perez Contreras
Tae M. Huh
ECE 118
April 26, 2025

Lab 3

Part 1:

```c
int main() {
  // ...
  while (1) {
    // Read potentiometer
    unsigned int pot_value = AD_ReadADPin(AD_PORTV5);

    // Set the Servo Position
    unsigned int pulse_time = 1000 + (pot_value * 1000) / 1023;
    printf("Pulse time = %d\r\n", pulse_time);

    // Set the Servo Pulse Time
    RC_SetPulseTime(RC_PORTV04, pulse_time);
    RC_GetPulseTime(RC_PORTV04);

    // Default: All LEDs off
    LED_SetBank(LED_BANK1, 0);
    LED_SetBank(LED_BANK2, 0);
    LED_SetBank(LED_BANK3, 0);

    // Decide which bank of LEDs to light up
    if (pot_value < LOW_THRESHOLD) {
      // Turn ON all LEDs in Bank 1
      LED_SetBank(LED_BANK1, 0x0F);
    } else if (pot_value < MEDIUM_THRESHOLD) {
      // Turn ON all LEDs in Bank 2
      LED_SetBank(LED_BANK2, 0x0F);
    } else {
      // Turn ON all LEDs in Bank 3
      LED_SetBank(LED_BANK3, 0x0F);
    }
  }
  // ...
}
```

Part 2:

```c
int main() {
  // ...
  while (1) {
    // Read potentiometer
    unsigned int pot_value = AD_ReadADPin(AD_PORTV5);

    // Map potentiometer value to PWM duty cycle (0-1000)
    unsigned int duty_cycle = (pot_value * 1000) / 1023;
    printf("DC Motor duty cycle = %d%%\r\n", duty_cycle/10);
    PWM_SetDutyCycle(PWM_MOTOR_PIN, duty_cycle);

    // Default: All LEDs off
    LED_SetBank(LED_BANK1, 0);
    LED_SetBank(LED_BANK2, 0);
    LED_SetBank(LED_BANK3, 0);
    // Calculate how many LEDs to light (0-12 LEDs for 0-3.3V)
    unsigned int num_leds = (pot_value * 12) / 1023;

    // Display LEDs across all three banks
    if (num_leds <= 4) {
      // Light 0-4 LEDs in Bank 1
      LED_SetBank(LED_BANK1, (1 << num_leds) - 1);
    } else if (num_leds <= 8) {
      // Bank 1 fully lit, light 0-4 LEDs in Bank 2
      LED_SetBank(LED_BANK1, ALL_LEDS_ON);
      LED_SetBank(LED_BANK2, (1 << (num_leds - 4)) - 1);
    } else {
      // Banks 1 & 2 fully lit, light 0-4 LEDs in Bank 3
      LED_SetBank(LED_BANK1, ALL_LEDS_ON);
      LED_SetBank(LED_BANK2, ALL_LEDS_ON);
      LED_SetBank(LED_BANK3, (1 << (num_leds - 8)) - 1);
    }
  }
  // ...
}
```

Part 4:

In this section we are intended to drive the motor by using a switch. To quickly switch between setups in the previous section we decided to not have a physical switch. Instead we manually hard coded a switch my setting a value to 0 or 1 (ON or OFF) in the code. Later we added a switch to verify to ourselves we could drive the signal with a switch. To implement this design we had a switch hooked up to 3.3V and then ground. We then fed the voltage from the negative side of the switch into the UNO32 stack. This was fairly easy but not implementing it helped us for checkoff.

```c
int main() {
  // ...
  while (1) {
    // Read potentiometer and switch
    unsigned int pot_value = AD_ReadADPin(AD_PORTV5);
    unsigned int switch_state = 1;

    // Map potentiometer value to PWM duty cycle (0-1000)
    unsigned int duty_cycle = (pot_value * 1000) / 1023;
    printf("DC Motor duty cycle = %d%%\r\n", duty_cycle/10);
    PWM_SetDutyCycle(PWM_MOTOR_PIN, duty_cycle);

    // Set direction based on switch state
    if (1) {
      // Direction 1
      IO_PortsWritePort(PORTZ, DIRECTION_PIN1);
      IO_PortsClearPortBits(PORTZ, DIRECTION_PIN2);
      printf("Direction: Clockwise\r\n");
    } else {
      // Direction 2
      IO_PortsClearPortBits(PORTZ, DIRECTION_PIN1);
      IO_PortsWritePort(PORTZ, DIRECTION_PIN2);
      printf("Direction: Counter-Clockwise\r\n");
    }

    // Default: All LEDs off
    LED_SetBank(LED_BANK1, 0);
    LED_SetBank(LED_BANK2, 0);
    LED_SetBank(LED_BANK3, 0);

    // Calculate how many LEDs to light (0-12 LEDs for 0-3.3V)
    unsigned int num_leds = (pot_value * 12) / 1023;
    /* LED Display for speed */
    if (num_leds <= 4) {
      LED_SetBank(LED_BANK1, (1 << num_leds) - 1);
    } else if (num_leds <= 8) {
      LED_SetBank(LED_BANK1, ALL_LEDS_ON);
      LED_SetBank(LED_BANK2, (1 << (num_leds - 4)) - 1);
    } else {
      LED_SetBank(LED_BANK1, ALL_LEDS_ON);
      LED_SetBank(LED_BANK2, ALL_LEDS_ON);
      LED_SetBank(LED_BANK3, (1 << (num_leds - 8)) - 1);
    }

  }
  // ...
}
```

Part 5:

Starting the motor was a bit of a difficult. The code ran but it seemmed as if the motor could only ever get half the radius. The motor would never do a complete revolutions. This made checking the step a little odd as it meant we had to distinguish between a half revolution and a stutter in the motor. To remove inaccuracies, we detached the motor from the base and held it in our hands. With this method we could feel stutters in the motors gears allowing for more accurate results than a visual analysis.

In order to properly understand the motors it is important to analyze at what step rates the motor can function and tolerate. To understand these bounds the motor was tested at a linear range from 200-800 with steps of 100. After getting the data for the steps at this range then we incremented by 25-50 to find more accurate values to where the motor would stutter.

Motor Setting	**Behavior**
200	bad stutter
225	ok Stutter
250	Almost no stutter (min step rate)
300	no stutter
400	no stutter
600	no stutter
700	no stutter
750	ok but minimal stutter
800	Very bad stuttering

```
int main() {
  // ...
  while (1) {
    // Read potentiometer and switch
    uint16_t pot_value = AD_ReadADPin(AD_PORTV5);
    uint8_t switch_state = 1;  // Default to FORWARD

    // Map potentiometer value to step delay (faster when pot is high)
    // Minimum delay is 1000/maxStepRate ms, maximum is 100ms
    uint16_t min_delay = 1000 / maxStepRate;
    uint16_t max_delay = 100;
```

```c
    stepDelay = max_delay - ((pot_value * (max_delay - min_delay)) / 1023);
    if (stepDelay < min_delay) stepDelay = min_delay;

    // Check if it's time to take a step
    if (TIMERS_GetTime() - lastStepTime >= stepDelay) {
      lastStepTime = TIMERS_GetTime();

      // Take a step in the direction determined by switch
      Stepper_Step(switch_state);

      // Increment step counter
      stepCounter++;
      if (stepCounter >= STEPPER_STEPS) {
        stepCounter = 0;
        printf("Completed full revolution\r\n");
      }
    }

    /* LED Display for speed and direction */
    // Default: All LEDs off
    LED_SetBank(LED_BANK1, 0);
    LED_SetBank(LED_BANK2, 0);
    LED_SetBank(LED_BANK3, 0);

    // Calculate how many LEDs to light based on speed (0-12 LEDs)
    uint8_t speed_percentage = (pot_value * 100) / 1023;
    uint8_t num_leds = (speed_percentage * 12) / 100;

    // Display speed on LEDs
    if (num_leds <= 4) {
      // Direction indication on top bit
      uint8_t led_pattern = (1 << num_leds) - 1;
      if (switch_state) {
        led_pattern |= 0x08;  // Set MSB for direction
      }
      LED_SetBank(LED_BANK1, led_pattern);
    } else if (num_leds <= 8) {
      // Bank 1 fully lit, light 0-4 LEDs in Bank 2
      uint8_t led_pattern = (1 << (num_leds - 4)) - 1;
      if (switch_state) {
        led_pattern |= 0x08;  // Set MSB for direction
      }
      LED_SetBank(LED_BANK1, ALL_LEDS_ON);
      LED_SetBank(LED_BANK2, led_pattern);
    } else {
      // Banks 1 & 2 fully lit, light 0-4 LEDs in Bank 3
      uint8_t led_pattern = (1 << (num_leds - 8)) - 1;
      if (switch_state) {
        led_pattern |= 0x08;  // Set MSB for direction
      }
      LED_SetBank(LED_BANK1, ALL_LEDS_ON);
      LED_SetBank(LED_BANK2, ALL_LEDS_ON);
      LED_SetBank(LED_BANK3, led_pattern);
    }
  }
  // ...
}
```

Part 6:

With the dedicate board the motor would now fully revolve. With this not we repeated the step rate test. With the dedicated board the motor seemed to get a larger step rate control range. For fun we included the potentiometer in the circuit allowing us to.

We are unsure if the issue in part 5 pertained as a code issue, motor, or H-Bridge issue. Many boards had broken bits and pieces and we didn't have time to diagonose which was the cause specificially.

```c
int main() {
  // ...
  while (1) {
    // Read potentiometer for speed control
    uint32_t pot_value = AD_ReadADPin(AD_PORTV5);

    // Use potentiometer to control step rate (50-1000 steps/sec)
    // uint32_t manual_rate = 50 + ((pot_value * 950) / 1023);
    uint32_t manual_rate = 700;
    Stepper_SetRate(manual_rate);
    Stepper_SetSteps(FORWARD, manual_rate)

    // Check if it's time to take a step (using non-blocking delay)
    if (TIMERS_GetTime() - lastStepTime >= (1000 / manual_rate)) {
      lastStepTime = TIMERS_GetTime();

      Stepper_SetSteps(switch_state, currentStepRate);
      Stepper_StartSteps();

      // Toggle direction every 100 steps
      (stepCounter)++;
      if (stepCounter >= 100) {
        stepCounter = 0;
        printf("Direction changed to: %s\r\n", switch_state ? "FORWARD" : "REVERSE");
      }

    }

    // LED Display calculation
    uint8_t num_leds = (manual_rate * 12) / 1000;
    updateLEDDisplay(num_leds);
  }
  // ...
}
```

Motor Setting	Behavior
200	bad stutter
225	ok Stutter
250	Almost no stutter (min step rate)
300	no stutter
400	no stutter
600	no stutter
700	no stutter
750	ok but minimal stutter

800	Very bad stuttering