Lab 4: Design and Implement a Finite State Machine

Group 5
Juliette Mitrovich, Sheila Moroney, and Sujani Patel

Lab Summary

The purpose of this lab was to design and implement a finite state machine for a two-floor elevator using an Arduino. We were able to apply what we learned about finite state machines in class to design our own. Using the known inputs and outputs, we determined the states we needed. We also learned how to implement a state machine in Arduino code. By using a combination of nested switches, we could keep track of all 16 input combinations. The physical elevator was represented by a servo that moves an arm 180 degrees. An optical interrupter was placed on either side to detect the "floor" the elevator was on. The optical interrupters were quick to implement after learning about them in the previous lab.

Finite State Machine Design

We started our design process by writing a truth table for the system. With given four input arguments, we knew from the equation that we would get 16 different combinations. After assessing the problem statement, we determined that we would have five states of output cases for our elevator to display: at first floor, moving up, at second floor, moving down, and error. To start our finite state machine, we drew these five states as circles with the output values according to those states. The three output values represent the LED for moving up, the LED for moving down, and the LED for an error.

To design the interactions of our states, we used our truth table to help figure out the inputs that would bring us to a specific state. For example, to get to the "moving up" state, the elevator must start at the first floor and the up button must be pressed. The second-floor sensor must be zero and it should not matter if the down button is pressed or not. This input is shown as "1X10" in Figure 1 below. After we determined the correct inputs to get us to each state, we determined the input combinations to keep the states the same. For example, while the elevator is moving to the second floor, we don't want any additional button pushes from either button to change the state. We show this as "XX00" in our diagram: pressing either button is not relevant and both floor sensors are zero. To finish our FSM, we needed to figure out which inputs at each state would bring us to the error state. We considered several possible error cases for our elevator. While the elevator is moving, up or down, and one or both floor sensors detect an obstruction, the elevator will stop, and the error LED will turn on. If the elevator is at a floor and the other floor sensor detects an obstruction, the elevator will remain still, and the error LED will turn on. If the error is solved (the obstruction is removed), it will leave the error state and the LED will turn off.

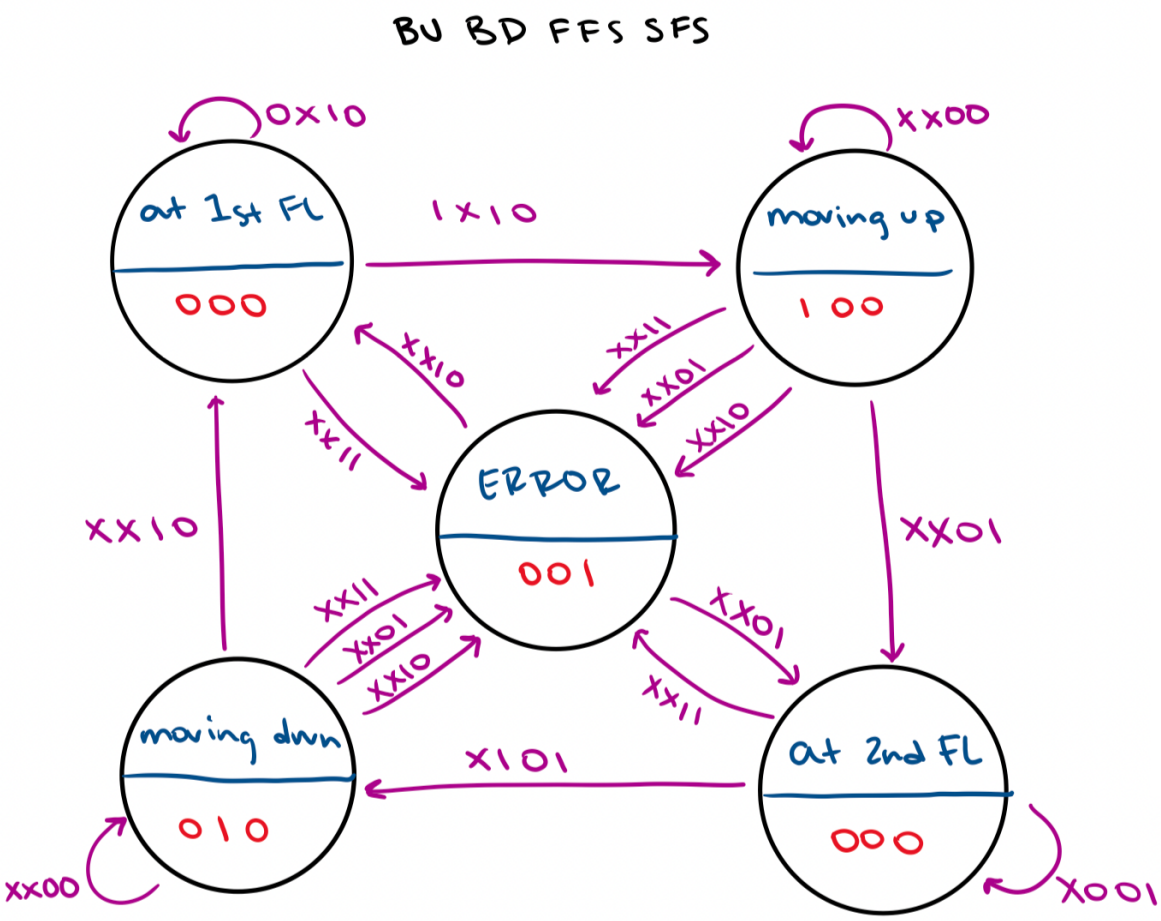

Figure 1. Diagram of the finite state machine

Translating the FSM to Code

To translate the finite state machine to Arduino code, we chose to mainly use the "switch" command because that is what was suggested in the Lab 4 description. Since there were four inputs we were tracking, we implemented nested switch cases. As a team, we worked together to write all the cases, commenting the code as we went so, we did not get lost in the code. Because the inputs were ordered as button up, button down, first floor sensor, and second floor sensor, we ordered the switch cases so they would resemble this. Once every state was defined, we went back through and printed to the serial monitor the state and what should be happening when it runs. We did this to make debugging the elevator easier when we got to the testing phase. Some of the error states were already defined from the 16 possible combinations of inputs. However, the situation where the first or second floor sensor is triggered before the elevator arrives to the desired floor was not accounted for. To do so, we added two more switch cases within the state "0000," which represents the elevator moving between floors. In this state, the switch tracks whether the FFS (first floor sensor) variable or the SFS (second floor sensor) variable is 1 or 0. If something blocks either sensor, making the variable display a 1, the elevator will stop moving, the up and down lights go off, and the error light goes on. If the sensor variable

remains a 0, the elevator continues moving as planned. Now that we had all the error states accounted for, we added the code to allow the inputs and outputs to run. For the push buttons controlling “up” and “down”, we simply added a line of code to read each button. When a button was pushed down, it would read a 1, and when it was not pushed, it would read a 0. The first and second floor sensor required slightly more code. Because they are analog devices, they read a number between 0 and 1023. When there is nothing in between the photo interrupter, it reads a value around 1000, and when there is something in between the photo interrupter, it reads a value around 80. These numbers are not constant and can vary slightly each time you block the device. The make sure the sensors could output a 1 or a 0, we created a function with if statements. That said, if the sensor value read is greater than 500, nothing is blocking the sensor, read 0. If the sensor value read is less than 500, something is blocking the sensor, read 1. The final step for the code was to add the servo movement. We defined position 0 as the first floor and position 180 as the second floor. Rather than using PWM, we chose to use the Arduino servo library to move the servo for simplicity. Now that the code was complete, it was time to wire everything up and test/debug the code.

Wiring and Testing

Before wiring the entire elevator, we wired each component separately to confirm our electronics were working. For the first optical interrupter, we used the same wiring setup that we used in the counting lab, a 560-ohm resistor on the LED side and a 50kOhm resistor on the emitter side of the transistor. For the second optical interrupter, we used the same 560-ohm resistor for the LED, but we did not have another 50kOhm resistor to use. To remedy this, we created a 50k Ohm resistor by connecting two 100k Ohm resistors in parallel. Both of those modules were wired into analog pins on the Arduino. We set up the buttons and the LEDs, each with a 220 Ohm resistor. For fun, we 3D printed clear arrow covers and alert covers for the LED to match their indicator purpose. The servo has a wire to ground and another to power. The third servo wire is the signal, which we connected to pin 2 on the Arduino. We mounted the servo between the optical interrupters on the breadboard. We used a 3D printed arm for the servo to set off the interrupts. The testing and debugging process went much smoother than expected. We only ran into minor issues such as the LEDs not turning on because the wrong variable had been written in the code. All in all, it was a very smooth process, and we were pleased with how seamlessly switch cases worked, since this was the first time any of us had used the command.

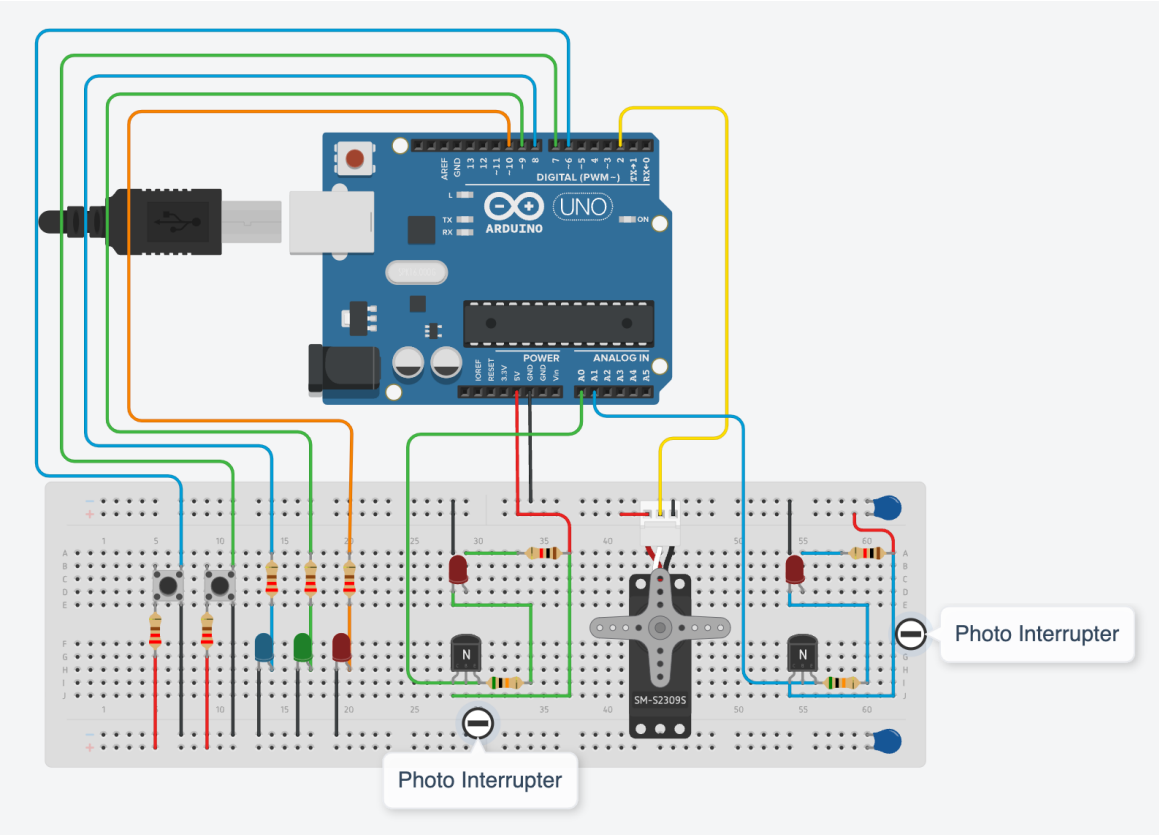

Figure 2. Final wiring diagram

Appendix A

```
/*
  ME 545 Lab 4 Group 5
  2 floor elevator finite state machine
  Juliette Mitrovich, Sheila Moroney, Sujani Patel
*/

///// LIBRARIES /////
#include <Servo.h>

///// INITIALIZE INPUT AND OUTPUT ARDUINO PINS /////
int buttonUp_pin = 6; // button to move up
int buttonDown_pin = 7; // button to move down

int indicatorUp = 8; // LED to indicate moving up
int indicatorDown = 9; // LED to indicate moving down
int indicatorErr = 10; // LED to indicate and error

int FFS_pin = A0; // first floor sensor (FFS) pin
int SFS_pin = A1; // second floor sensor (SFS) pin

///// INITIALIZE VARIABLES TO STORE INPUTS AND OUTPUTS /////
int buttonUp;
int buttonDown;

int FFS;
int SFS;

///// INITIALIZE SERVO MOTOR /////
Servo myServo;  // create servo object to control a servo
int pos = 0;    // variable to store the servo position

///// FUNCTIONS /////
// make FFS output read 1 or 0
void readFFS() {
  FFS = analogRead(FFS_pin);
  if (500 < FFS) {
    FFS = 0;
  }
  else {
```

```
    FFS = 1;
  }
}

// make SFS output read 1 or 0
void readSFS() {
  SFS = analogRead(SFS_pin);
  if (500 < SFS) {
    SFS = 0;
  }
  else {
    SFS = 1;
  }
}

void setup() {
  Serial.begin(9600); // turn on serial monitor

  // set input pins
  pinMode(buttonUp_pin, INPUT);
  pinMode(buttonDown_pin, INPUT);
  pinMode(FFS_pin, INPUT);
  pinMode(SFS_pin, INPUT);

  // set output pins
  pinMode(indicatorUp, OUTPUT);
  pinMode(indicatorDown, OUTPUT);
  pinMode(indicatorErr, OUTPUT);

  myServo.attach(2);  // attaches the servo on pin 2 to the servo object
  myServo.write(pos); // set the servo to position 0 when powered on

}

void loop() {
  // contantly read the buttons and the floor sensors
  buttonUp = digitalRead(buttonUp_pin);
  buttonDown = digitalRead(buttonDown_pin);
  readFFS();
  readSFS();
```

```
switch (buttonUp) {
  case 0: // button up case 0
    // watch button down
    switch (buttonDown) {
      case 0: // button down
        // watch first floor sensor
        switch (FFS) {
          case 0: // ffs
            // watch second floor sensor
            switch (SFS) {
              case 0: // sfs (0000)
                Serial.println("0000: moving floors");
                // watch first floor for interruptions
                switch (FFS) {
                  case 0:
                    // do nothing
                    break;

                  case 1:
                    // something has blocked the floor before you got there
                    // stop the servo (elevator) and turn on error light
                    digitalWrite(indicatorErr, HIGH);
                    digitalWrite(indicatorUp, LOW);
                    digitalWrite(indicatorDown, LOW);
                    myServo.detach();
                    break;

                }
                break;
                // watch second floor for interruptions
                switch (SFS) {
                  case 0:
                    // do nothing
                    break;

                  case 1:
                    // something has blocked the floor before you got there
                    // stop the servo (elevator) and turn on error light
                    digitalWrite(indicatorErr, HIGH);
```

```
            digitalWrite(indicatorUp, LOW);
            digitalWrite(indicatorDown, LOW);
            myServo.detach();
            break;
          }
          break;
          break;

        case 1: // sfs (0001)
          digitalWrite(indicatorErr, LOW); // turn error light off if it was on
          digitalWrite(indicatorUp, LOW); // turn up light off
          Serial.println("0001: at second floor or, coming out of error");
          break;
      }
      break;

    case 1: // ffs
      // watch second floor sensor
      switch (SFS) {
        case 0: // sfs (0010)
          digitalWrite(indicatorErr, LOW); // turn error light off if it was on
          digitalWrite(indicatorDown, LOW); // turn down light off
          Serial.println("0010: down button pressed, move down, or coming out of error");
          break;


        case 1: // sfs (0011)
          digitalWrite(indicatorErr, HIGH); // turn on error light
          digitalWrite(indicatorDown, LOW); // turn off down light if it was on
          digitalWrite(indicatorUp, LOW); // turn off up light if it was on
          Serial.println("0011 error: both floors");
          break;
      }
  }
  break;


case 1: // button down
  // watch first floor sensor
  switch (FFS) {
```

```
      case 0: // ffs
        // watch second floor sensor
        switch (SFS) {
          case 0: // sfs (0100)
            Serial.println("0100: button press while moving, stay in state");
            break;

          case 1: // sfs (0101)
            digitalWrite(indicatorDown, HIGH);
            digitalWrite(indicatorErr, LOW); // turn error light off if it was on
            myServo.attach(2);
            myServo.write(0); // move to the down (first floor) position
            Serial.println("0101: down button pressed, move down, or coming out of error");
            delay(500);
            break;
        }
        break;

      case 1: // ffs
        // watch second floor sensor
        switch (SFS) {
          case 0: // sfs (0110)
            digitalWrite(indicatorErr, LOW); // turn error light off if it was on
            Serial.println("0110: down button pressed, move down, or coming out of error");
            break;

          case 1: // sfs (0111)
            digitalWrite(indicatorErr, HIGH);
            digitalWrite(indicatorUp, LOW);
            digitalWrite(indicatorDown, LOW);
            Serial.println("0111 error: both floors and button down");
            break;
        }
        break;

    }
    break;

}
break;
```

```
case 1: // button up
  // watch button down
  switch (buttonDown) {
    case 0: // button down
      // watch first floor sensor
      switch (FFS) {
        case 0: // ffs
          // watch second floor sensor
          switch (SFS) {
            case 0: // sfs (1000)
              Serial.println("1000: button pressed while moving, don't care");
              break;

            case 1: // sfs (1001)
              digitalWrite(indicatorErr, LOW); // turn error light off if it was on
              Serial.println("1001: up pressed at 2nd floor, don't move");
              Serial.println("or, coming out of error");
              break;

          }
          break;

        case 1: // ffs
          // watch second floor sensor
          switch (SFS) {
            case 0: // sfs (1010)
              digitalWrite(indicatorUp, HIGH);
              digitalWrite(indicatorErr, LOW); // turn error light off if it was on
              myServo.attach(2);
              myServo.write(180); // move to the up (second floor) position
              Serial.println("1010: on 1st and pressed up, move up");
              delay(500);
              break;

            case 1: // sfs (1011)
              digitalWrite(indicatorErr, HIGH);
              digitalWrite(indicatorUp, LOW);
              digitalWrite(indicatorDown, LOW);
              Serial.println("1011 error: both floors and button up");
```

```
        break;
      }
      break;

  }
  break;


case 1: // button down
  // watch first floor sensor
  switch (FFS) {
    case 0: // ffs
      // watch second floor sensor
      switch (SFS) {
        case 0: // sfs (1100)
          Serial.println("1100: button pressed when moving, don't care");
          break;

        case 1: // sfs (1101), even though you pressed both buttons, still move down
          digitalWrite(indicatorDown, HIGH);
          digitalWrite(indicatorErr, LOW); // turn error light off if it was on
          myServo.attach(2);
          myServo.write(0); // move to the down (first floor) position
          Serial.println("1101: down button pressed, move down or, coming out of error");
          delay(500);
          break;
      }
      break;

    case 1: // ffs
      // watch second floor sensor
      switch (SFS) {
        case 0: // sfs (1110), even though you pressed both buttons, still move up
          digitalWrite(indicatorUp, HIGH);
          digitalWrite(indicatorErr, LOW); // turn error light off if it was on
          myServo.attach(2);
          myServo.write(180); // move to the up (second floor) position
          Serial.println("1110: on 1st and pressed up, move up");
          delay(500);
```

```
            break;

          case 1: // sfs (1111)
            digitalWrite(indicatorErr, HIGH);
            digitalWrite(indicatorUp, LOW);
            digitalWrite(indicatorDown, LOW);
            Serial.println("1111 error: both floors and button up");
            break;
        }
        break;
      }
      break;
    }
    break;
  }
}
```