ACTIVITY 6B - Arduino 6B: Finite-State Machine (Washing Machine - Selectable Wash Duration)

Jason Harvey Lorenzo, March 16, 2024
ENGG 122.02, Department of Electronics, Computer, and Communications Engineering
School of Science and Engineering, Ateneo de Manila University
Quezon City, 1108, Philippines
jason.lorenzo@obf.ateneo.edu

***Abstract*—This paper presents the design and implementation of a Finite State Machine (FSM) to emulate the operation of a basic washing machine using an Arduino microcontroller. The FSM accurately replicates the sequential stages of a washing cycle, including filling water, washing, draining, rinsing, and pausing, with adjustable wash durations. Utilizing LEDs for visual feedback and push-button inputs for user control, the system demonstrates the effectiveness of FSMs in modeling and controlling discrete-state systems. Additionally, recommendations are provided to enhance the efficiency and usability of the system, including adjusting time delays, optimizing user inputs, and streamlining code structure for improved performance.**

***Index Terms*—Arduino, Finite State Machine (FSM), washing machine simulation**

I. Introduction

This project entails creating and deploying a Finite State Machine (FSM) to replicate the functionality of a basic washing machine, complete with adjustable wash durations. To simulate the various stages of the washing process, an LED shield will be utilized. Further details regarding the FSM's specifications are provided below:

- The flow of the wash cycle goes as follows:
  1) Idle
  2) Fill Water
  3) Wash
  4) Drain1
  5) Refill Water
  6) Rinse
  7) Drain2
  8) Return to Idle
- The time intervals between cycles are standardized to 5 seconds, with the exception of two transitions: the duration between IDLE and FILL WATER, which relies on the moment the START/PAUSE button is pressed, and the interval between Wash and Drain1, which depends on the selected Mode.
- In the IDLE state, users can select from three distinct wash modes, each associated with its respective wash duration, using a button (SET WASH MODE):
  1) Quick - 3 seconds
  2) Normal - 5 seconds
  3) Delicate - 8 seconds

  The default wash mode will be Normal.
- Once a wash mode is selected, the user can START the wash cycle using a button (START/PAUSE).
- The wash cycle operation can be paused in all states except for the IDLE state. When paused, the operation temporarily transitions to the PAUSE state and will only resume when the START/PAUSE button is pressed again. Resuming operation after pausing does not reset the duration of the state; rather, the operation continues from where it was paused.
- An EMERGENCY STOP (RESET) button can be pressed at any state, stopping the entire wash cycle and returning to the IDLE state.
- If none of the conditions are met, the implicit self-loop of the system continues.

The LED shield will present the current state/input by turning on corresponding LEDs as follows:

TABLE I
LED Functions

LED	Function
D11	Quick Wash Mode
D12	Normal Wash Mode
D13	Delicate Wash Mode
D10	Idle State
D9	Fill Water State
D8	Wash State
D7	Drain1 State
D6	Re-fill Water State
D5	Rinse State
D4	Drain2 State
D14	Pause State (blinking)

II. Program Listing

A. Source Code

```
// state names for better readabilty
const int IDLE = 0;
const int FILL = 1;
const int WASH = 2;
const int DRAIN1 = 3;
const int REFILL = 4;
const int RINSE = 5;
const int DRAIN2 = 6;
const int PAUSE = 7;
```

```
int countT = 5;

volatile int state = IDLE;
volatile int T = 1;
int washDuration = 5;

volatile int activeLED = 10;

// checkpoint to go back from PAUSE
volatile int state_holder;

bool SHIFT_MODE;

void setup() {
  pinMode(A5, INPUT_PULLUP);
  for (int i = 4; i <= 13; i++)
    pinMode(i, OUTPUT);
  // Normal Mode initially
  digitalWrite(12, HIGH);
  attachInterrupt(0, start_pause,
    FALLING);
  attachInterrupt(1, reset, FALLING);
}

void start_pause() {
  if (state == IDLE) {
    state = FILL;
    digitalWrite(activeLED, LOW);
    T = 0;
  } else if (state == PAUSE) {
    state = state_holder;
    digitalWrite(activeLED, LOW);
  } else {
    state = PAUSE;
    digitalWrite(activeLED, LOW);
  }
}

void reset() {
  state = IDLE;
  digitalWrite(activeLED, LOW);
}

void loop() {
  SHIFT_MODE = !digitalRead(A5);

  switch (state) {
    case IDLE:
      activeLED = 10;
      // show current state
      digitalWrite(activeLED, HIGH);
      if (SHIFT_MODE) {  // A5
        if (washDuration == 3) {
          washDuration = 5;
          digitalWrite(11, LOW);
          digitalWrite(12, HIGH);
        } else if (washDuration == 5) {
          washDuration = 8;
          digitalWrite(12, LOW);
          digitalWrite(13, HIGH);
        } else if (washDuration == 8) {
          washDuration = 3;
          digitalWrite(13, LOW);
          digitalWrite(11, HIGH);
        }
      }
      break;

    case FILL:
      activeLED = 9;
      digitalWrite(activeLED, HIGH);
      state_holder = state;
      if (T == countT) {
        state = WASH;
        digitalWrite(9, LOW);
        T = 0;
      }
      break;

    case WASH:
      activeLED = 8;
      digitalWrite(activeLED, HIGH);
      state_holder = state;
      if (T == washDuration) {
        state = DRAIN1;
        digitalWrite(8, LOW);
        T = 0;
      }
      break;

    case DRAIN1:
      activeLED = 7;
      digitalWrite(activeLED, HIGH);
      state_holder = state;
      if (T == countT) {
        state = REFILL;
        digitalWrite(7, LOW);
        T = 0;
      }
      break;

    case REFILL:
      activeLED = 6;
      digitalWrite(activeLED, HIGH);
      state_holder = state;
      if (T == countT) {
        state = RINSE;
        digitalWrite(6, LOW);
        T = 0;
      }
      break;
    case RINSE:
      activeLED = 5;
      digitalWrite(activeLED, HIGH);
      state_holder = state;
      if (T == countT) {
        state = DRAIN2;
        digitalWrite(5, LOW);
        T = 0;
      }
      break;

    case DRAIN2:
      activeLED = 4;
      digitalWrite(activeLED, HIGH);
      state_holder = state;
      if (T == countT) {
        state = IDLE;
        digitalWrite(4, LOW);
        T = 0;
      }
      break;

    case PAUSE:
      activeLED = 14;
      digitalWrite(activeLED, HIGH);
      T--;
      // Ensures current duration remains
      break;
  }
  T++;
  delay(500);
  // to blink when on PAUSE
  digitalWrite(3, LOW);
  delay(500);
}
```

B. How It Works

This code implements a simple finite state machine (FSM) to simulate the operation of a washing machine. The system has several states (IDLE, FILL, WASH, DRAIN1, REFILL, RINSE, DRAIN2, PAUSE) represented by integer constants.

In the setup function, pins are initialized for input (A5) and output (pins 4 to 13). An interrupt is attached to both 2 (starting or pausing the machine) and pin 3 (for resetting the machine).

The loop function continuously checks the current state of the machine and performs actions accordingly. Depending on the state, different LEDs are activated to indicate the current operation. For example, when in the IDLE state, the LED corresponding to pin 10 is lit. The program reads the status of the shift button (A5) to adjust the wash duration. In every iteration of this loop function, T is incremented. This is the normal condition to transition between states. Each state runs until 5 seconds or the set wash duration (for WASH state) is reached and the machine moves to the next state. Note that user input can interrupt this normal functioning as discussed below.

The start_pause function handles the logic for starting or pausing the washing cycle. If the system is in the IDLE state, pressing the start button transitions to the FILL state. If it's in the PAUSE state, it resumes from where it left off, otherwise, it pauses the cycle.

The reset function resets the system to the IDLE state when the reset button is pressed. Note that the start_pause and reset functions are interrupt handlers, which means they can be invoked at any time when their corresponding buttons are pressed. Upon execution, these functions interrupt the normal flow of the program, executing their designated tasks. Once completed, the program resumes its operation from the state it was in before the interruption occurred.

III. Design Process and Discussion

A. Background Theory

Finite State Machines (FSMs) are widely used in various fields of engineering and computer science for modeling and controlling systems with discrete states and transitions. In the context of embedded systems and microcontroller programming, FSMs offer an effective approach to designing complex systems with multiple operational states.

In this experiment, we employed an Arduino microcontroller to implement an FSM simulating the operation of a simple washing machine. The system consists of several operational states, including IDLE, FILL, WASH, DRAIN1, REFILL, RINSE, DRAIN2, and PAUSE, each representing a distinct phase of the washing cycle. Transitions between these states are triggered by specific events, such as button presses or predefined time intervals.

B. Design Process and Discussion

The design process for implementing the washing machine FSM involved several key steps:

1) **State Identification:** First, the distinct states that characterize the washing machine's operation were identified. These states were chosen based on the typical phases of a washing cycle, such as filling, washing, draining, rinsing, and pausing.
2) **Event and Transition Definition:** Next, the events that trigger transitions between states were defined. These events include user inputs, such as button presses to start, pause, or reset the washing cycle, as well as predefined time intervals to simulate the duration of each phase.
   The output of the first two steps is the following FSM Diagram in Fig. 1.
   The boolean conditions are:
   - WD = SET WASH DURATION pressed (setting wash time duration to 3, 5, or 8)
   - R = RESET pressed
   - S/P = START/PAUSE pressed
   - T = 5 seconds has elapsed
   - WT = the set wash duration time has elapsed

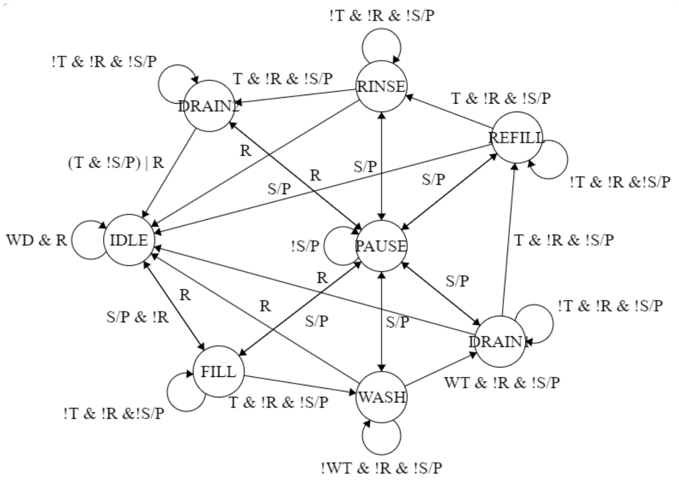

Fig. 1. Washing Machine with Selectable Wash Duration FSM Diagram

Note that whenever a state becomes active, the information about the current state is stored in a variable called state_holder. This ensures that when the washing machine enters the PAUSE state (triggered by pressing the START/PAUSE button), it retains the information about the previous state. Consequently, when the START/-PAUSE button is pressed again to resume operation, the washing machine knows which state to return to.

3) **Hardware Setup:** As seen in Fig. 2, the hardware setup comprised an Arduino microcontroller connected to a laptop for both power supply and program uploading. Positioned atop the Arduino was an LED shield, its LEDs function according to Table I. The A5 button on this shield served as the SET WASH DURATION button. To enable the utilization of interrupt pins for RESET and START/PAUSE inputs, two buttons, each with its pullup resistor, were incorporated.

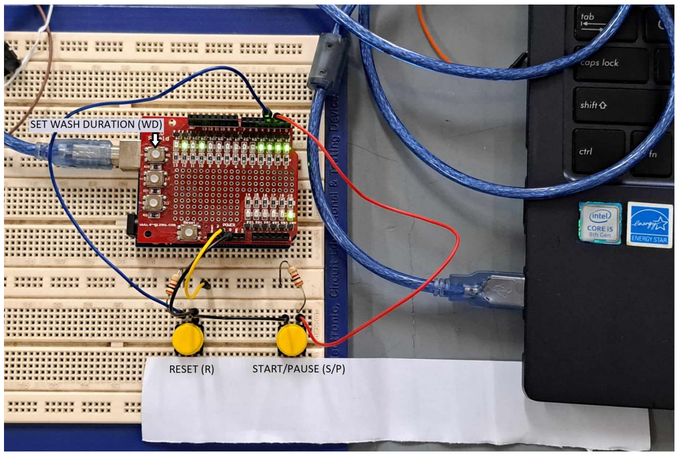

Fig. 2. Arduino Setup

4) **Testing and Validation:** Finally, the implemented FSM was tested to ensure that it correctly emulates the behavior specified in the introduction. This involved simulating various scenarios, such as starting, pausing, and resetting the washing cycle, as well as handling unexpected inputs or errors.

The implemented FSM successfully simulates the operation of a simple washing machine, demonstrating the effectiveness of FSMs for modeling and controlling discrete-state systems. By employing an Arduino microcontroller and simple input/output devices, a functional prototype that mimics the behavior of a real washing machine was successfully created.

However, there are several limitations to consider. The FSM implemented in this experiment is relatively basic and may not fully capture all the nuances of a real washing machine's operation. Additionally, the use of push-button inputs and LEDs for user feedback may be simplistic compared to modern washing machine interfaces, which often feature digital displays and touch-sensitive controls.

IV. Conclusion and Recommendation

The implementation of the Finite State Machine (FSM) to simulate the operation of a washing machine using an Arduino microcontroller proved successful. Through careful design and programming, we were able to replicate the sequential states of a washing machine cycle, including idle, filling water, washing, draining, rinsing, and pausing. The use of LEDs as indicators for different states provided the necessary visual feedback. Additionally, the incorporation of interrupt functions for pause and reset buttons added flexibility and user control to the system.

To enhance the efficiency and responsiveness of the washing machine simulation system, several recommendations are proposed. Firstly, adjusting the time delays for different states could better reflect realistic washing machine operation, with shorter durations for tasks like filling water and draining cycles. Secondly, converting the SET WASH DURATION button to an interrupt-driven input would streamline user interaction, eliminating the need to hold down the button and enhancing usability. Finally, optimizing the code structure to eliminate redundancy and streamline processes can improve memory usage and execution speed, leading to a more efficient and reliable system overall.