A Variable AC Power Source Based on the Arduino Platform Using Finite State Machines

[1]Muhammet Kenan Akinci and *[1]Suayb Cagri Yener
[1] Faculty of Engineering, Department of Electrical and Electronics Engineering Sakarya University, Turkey

Abstract

In this paper, we design an FSM (Finite State Machine) based controlling system by using Arduino board for a variable AC (alternative current) power source which is physically controlled with a variac attached to a DC Motor. We construct a finite state machine based control methodology by using an Arduino board. We analyze its advantages and disadvantages by comparing to basic control algorithms formed by program delays and branching statements. In the presented FSM based methodology, transition between states are achieved as an event trigger or condition change like a user input which is setting the desired voltage value in the variac control system. Experimental results show that finite state machine based design is faster and more stable than traditional conditional algorithms. Also it has been shown that the FSM based structure is realized effectively by using an Arduino board and it can provide new opportunities to digital control design.

Key words: FSM, Arduino, microcontroller board, AC power source, control algorithm, variac

1. Introduction

In today's world, new and cheaper control instruments are developed day by day. One of the current control equipment and very popular one is the Arduino board. Arduino is an open source microcontroller based development board with basic I/O (input/output) capabilities. It can be programmed on almost every operating system with the Processing Language [1]. It has an Atmel Atmega CPU (central processing unit) and digital I/O pins in addition to analog I/O pins. The board has its own unique design and ready-to-go for electronics prototyping. Arduino Uno has 8-bit Atmega328 CPU and it is the most basic Arduino board among other powerful counterparts such as Arduino Galileo or Arduino Due [1]. Arduino Uno model, has 14 digital I/O (six of them can be used as analog output) and 6 analog input pins. Thanks to the fact that Arduino is a low-cost platform, it is fairly popular in these days to use it as a prototyping platform for designing basic control systems.

*Corresponding author: Address: Faculty of Engineering, Department of Electrical and Electronics Engineering Sakarya University, 54187, Sakarya TURKEY. E-mail address: syener@sakarya.edu.tr, Phone: + 902642955826 Fax: + 902642955601

Finite state machine is an abstract machine has a finite number of states. It can only be in one of the states at any time and transition between these states are done by using event changes or triggering an event with a user input. In an FSM; the process is controlled by state identifiers at any state's last block of the code. Thanks to these identifiers, the program knows the current and next state itself. This control algorithm implements a basic but powerful technique for controlling structures. It is convenient because the program or code has a finite and fully controllable number of states. Designing an FSM with Arduino is simply creating a switch-case statement and using state identifiers to control current and next states.

There are several studies in the literature on digital applications by using Arduino boards. Arduino is used as a communicating tool for dynamic resistance measurement at high voltage circuit breakers by De Souza et al [2]. Zhuge Yan and Che used the wireless capabilities of Arduino to implement a remote image sensing system [3]. Shajahan and Anand studied with Arduino to design a data acquisition and control platform working on Android operating system [4]. Munadi and Akbar used Arduino and Matlab to control a servo motor by using fuzzy logic control algorithm [5].

There are also a lot of studies in the literature done by using finite state machines. Pedroni gives detailed and comprehensive information to theory of designing finite state machines for hardware level and describes design problems by using VHDL (VHSIC Hardware Description Language) and Verilog hardware description languages [6]. Gill studied about using multiple FSM structures together named "cascaded finite state machines" [7]. Chang used Finite State Machines on execution of fuzzy programs and showed that using FSM structures can be effective at different ways of executing fuzzy programs [8]. Chu et al. proposed an efficient and new state assignment technique for asynchronous finite state machines under the unbounded gate delay model [9]. Chen and Lin used finite state machines to model discrete event systems and gave an example that discrete event applications can be represented efficiently using FSM [10]. Kalay et al. studied about testing finite state machines based on XOR (Exclusive or) logic and used different type of logical structures such as D-type, JK-type flip flops on the testability of the finite state machines [11]. Rietsche presented a state assignment algorithm using T-type flip flops and showed that the area requirements of the finite state machine realization decreases if T-type flip flops are used [12].

The organization of the paper as follows; after this introduction, in the second section, the methodology and design principles of the system are described. In the third section, the FSM design and the whole system are described. FSM design is compared to other basic controlling methods such as if-else structures. Finally in the fourth section the study is finalized, showing that, based on FSM, Arduino can be successfully used to design control systems.

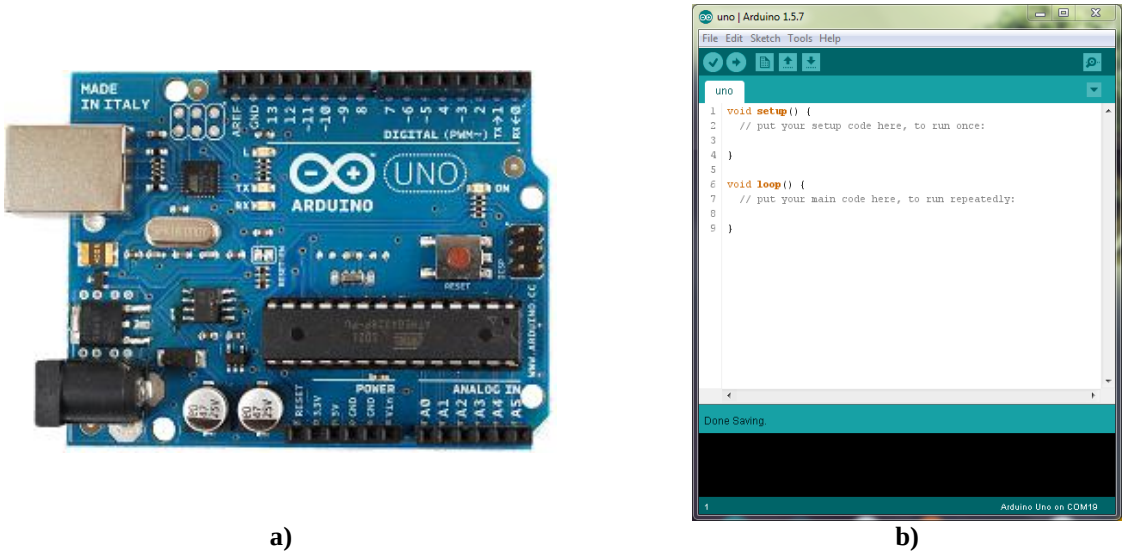
2. Methodology and Design

2.1. Arduino Microcontroller Board

In our AC Power Source System, Arduino Uno microcontroller board have been used. Arduino Uno has enough number of the digital and analog I/O pins for many basic applications such as

driving DC (direct current) motors, servos, relays or wireless communication application.

As the main motivation of the design, we have used digital I/O pins of the Arduino Uno board and also used L298 motor driver and basic circuitry for controlling the DC motor of the AC Power Source. Arduino Uno board is shown in Figure 1.a. The software part of the system is also developed using the Arduino IDE (Integrated Development Environment) which is simple and easy to use. The Arduino IDE is shown in Figure 1.b.



a) **b)**

Figure 1. a) Arduino Uno and b) Arduino IDE

2.2. Finite State Machine

Traditional programming structures such as if-else and basic switch-case are used for simple control applications. Delay functions are also used for time delays or waiting for an action to be executed. The delay functions make the microprocessor busy. Therefore no other actions can be executed at a time. Moreover in these methods the program does not know which situation it is. If some calculation or decision goes wrong, the whole system may collapse.

In FSM implementations, the program knows where and which situation it is because there is only a finite number of states. The program can't go outside of these states. If no calculation or no input occurs, the program remains in the current state which is named "idle". It is not necessary but in an FSM based structure, the program should return to idle state after completing the desired operation.

We have designed an FSM with a couple of states to control our system. First, the system resets itself at the beginning. Then user selects the calibration option which is the "calibration" state. In this state, the mechanical part of the AC Power Source, shaft, turns forwards and backwards once. Then Arduino calculates the total required time to reach from beginning to end of the AC Power Source. After this operation the calibration state is done, time is calculated and program goes back to idle state. Then user decides a voltage level by pressing the push buttons. This

decision is the start of the "operation" state. DC motor turns the shaft to reach desired voltage level. After reaching to the voltage level, operation state is done and program turns back to the idle state again. While there is a transition between states or program is in the operation state, if user makes another input, the system is capable to notice that input and make actions. For example, user can set another voltage level. The flow chart of this system is illustrated in Figure 2.

In Figure 2, the Switch1 and Switch2 states are used for the system's stability. If the shaft tries to go beyond start and the end point of the AC Power Source which are the maximum and minimum voltage value, the program recognizes the situation and goes to the Switch1 or Switch2 states. These states simply stops the motor. Then program goes to the idle state.

To achieve this control flow, switch-case structure are used to program the Arduino board. The state identifiers mentioned before are simply two variables stored in the main program. After each operation, the identifiers' values are changed to next state to notify the main program.

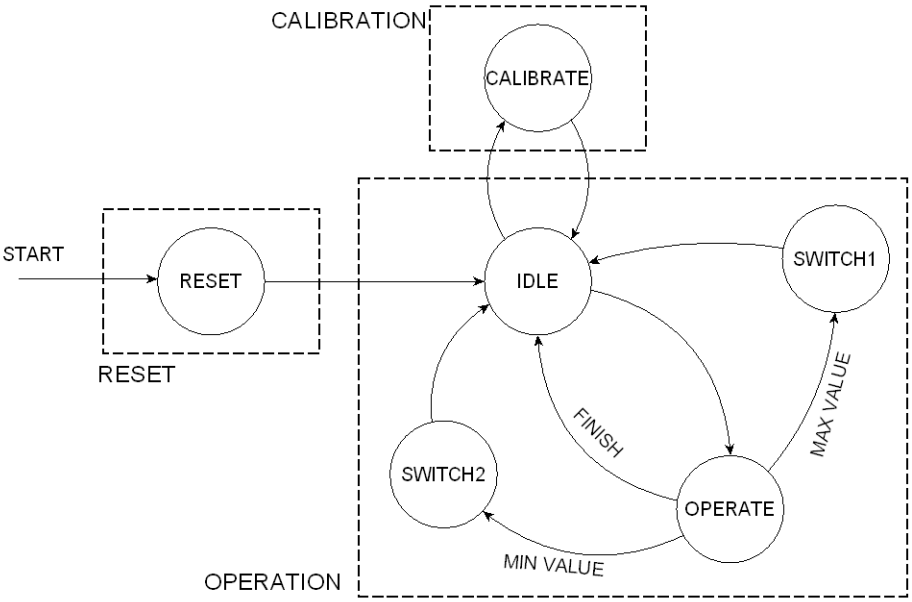


Figure 2. FSM based flowchart of the system

3. Design of the System

3.1. The Variable AC Power Source

The variable AC Power Source is based on a variac, controlled with a DC motor attached to its shaft. Two mechanical switches at the start and end point on the mechanical part of the AC Power Source is used for overturn control of the motor. The diagram of the whole system is

shown in Figure 3.

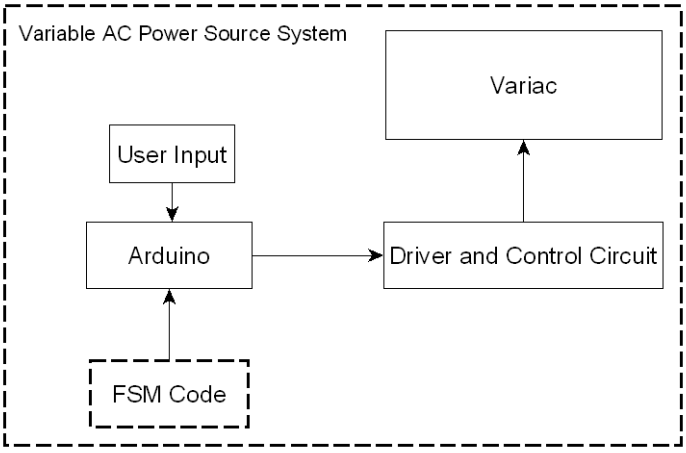


Figure 3. Block diagram of the AC Power Source system

There are 3 main states in the Variable AC Power Source system: the reset, the calibration and the operation.

3.2. States of the Operation

3.2.1. The Reset

At the beginning of the system when the Arduino and the motor driver circuitry boards powered up, the program resets the DC motor's position by using hardware interrupts of the Arduino Uno board that connected to switches. Through the reset, program will know the exact position of the DC motor so that in the next states, it can start operating from beginning of the motor's position.

3.2.2. The Calibration

The main objective of the calibration state is to calculate the required time for the shaft to go from the start point of the AC Power Source to its end point in the system. Calibration state has its own piece of code in itself. The whole process is done only once at the system's startup after the reset state and then the calculated time value is stored in the program memory.

At the beginning of the calibration, the DC motor's shaft is at the start point of the variac. It goes all the way to the endpoint, touches the Switch-2's contacts and closes them. The interrupt warns the main program for reaching the end point. Then the main program turns the motor backwards to the start point. The whole operation is done for precise calculation of the time.

The time value T_1 is the shaft's rotation from start to end point and the time value T_2 is the shaft's rotation from end to start point. Therefore the final time value "T_{avg}" can be calculated by taking average of these two values in Eq. (1):

$$T_{avg} = (T_1 + T_2) / 2 \quad (1)$$

This calculated T_{avg} value will be used to determine the required turn time of the motor's shaft for a desired voltage value. The pseudocode for the calibration state is given in Table 1.

Table 1. The Pseudocode for the Calibration State

```
STATE = Calibration;
     TURN Motor = Forwards;
     Start Counting Time;
     IF Switch2 is closed
          T1 = Estimated Time;
          TURN Motor = Backwards;
          IF Switch1 is closed
               STOP Motor;
               T2 = Estimated Time – T1;
     CALCULATE Tavg = (T1 + T2) / 2;
     PRINT "T";
STATE = IDLE;
```

After the calibration state, the program goes to the idle state and loops there. In other words; it is waiting for a user input.

3.2.3. The Operation

The operation state is the most important part of the main program. While the program is waiting in the idle state, user makes an input. Then the program jumps to the operation state and calculates the required time for turning the motor for the required voltage value.

At the beginning of the operation state, program calculates the required time by using Eq. (2).

$$T_{req} = ((| V_{user} - V_{actual} |).T_{avg}) / V_{max} \quad (2)$$

In Eq. (2), V_{max} is the maximum output value for the AC Power Source. V_{user} is the voltage value that user gives to the program by using input buttons. V_{actual} is the actual value of the voltage at the beginning of the state which is 0V. $V_{user} - V_{actual}$ can be negative value due to user's voltage selection. For negative values, we take the absolute value of the equation so that the required T_{req} value stays positive. If the V_{user} is smaller than V_{actual}; the motor will turn backwards and vice versa. After calculating the required T_{req}, the motor starts to turn as much as T_{req} value and stops. Therefore the desired voltage value is supplied.

The whole process is done in the operation state and after the program completes the operation, it goes back to the idle state for another user input which can be changing the voltage value again. The whole process is going to be done again for the new voltage value.

The switches mentioned above at the shaft's start and end point are the safety switches to disable overturning the DC motor. They are also controlled in the calibration and operation state of the program. In operation state, if one of that switches closes, the program jumps to the Switch1 or Switch2 states. These two identical states immediately makes the motor stop. Then program goes back to the idle state.

The pseudocode for the whole operation state is given in Table 2.

Table 2. The Pseudocode for the Operation State

```
STATE = Operation;
     CALCULATE Treq = ((Vuser- Vactual) * Tavg) / 220;
     Start Counting Time;
          TURN Motor FORWARDS or BACKWARDS;
          IF Estimated Time = Treq
                    STOP Motor;
                    STATE = IDLE;
          IF Switch2 = Closed
                    STOP Motor;
                    STATE = IDLE;
          IF Switch1 = Closed
                    STOP Motor;
                    STATE = IDLE;
```

4. Conclusions

In this paper, we have designed an FSM based AC power source controlling system by using Arduino board. We have shown that Arduino is a good option for implementing an FSM based control flow. Using Arduino and FSM, we have implemented a low-cost, fast and easy-to-use variable AC power source design. Also it has been shown that the Finite State Machine algorithm is reliable and fast for system operation. Thanks to finite state machine structure, our system is robust as expected due to the finite number of states and transitions between them. While operating, it is more responsive than the traditional control algorithms because our design is not using program delays which are blocking the microprocessor. User can interact with the program in any time of operation even while the system is operating. Also our states are simple and easy to maintain so that any improvement or change in the main program can be successfully implemented in the future.

References

[1] Banzi M. Getting Started with Arduino. 2nd ed. California: O'reilly Media; 2011.
[2] de Souza RT, Guedes da Costa E, de Araújo J.F, de Macedo E.C.T. A system for dynamic contact resistance with Arduino platform on MV and HV circuit breaker. Instrumentation and Measurement Technology Conference (I2MTC) Proceedings IEEE 2014; 369 – 373.
[3] Zhuge Yan, Che C. Remote image sensing platform based on Arduino. Computer Science and Electronic Engineering Conference (CEEC), 2014; 29 – 34.
[4] Shajahan A.H, Anand A. Data acquisition and control using Arduino-Android platform: Smart plug. Energy Efficient Technologies for Sustainability (ICEETS), 2013; 241 – 244.
[5] Munadi, Akbar M.A. Simulation of fuzzy logic control for DC servo motor using Arduino based on MATLAB/Simulink. Intelligent Autonomous Agents, Networks and Systems (INAGENTSYS), IEEE 2014; 42 – 46.
[6] Pedroni Volnei A. Finite State Machines in Hardware: Theory and Design (with VHDL and SystemVerilog), 1st ed. Massachusetts: The MIT Press; 2013.
[7] Gill A. Cascaded Finite-State Machines. Electronic Computers, IRE Transactions (Volume:EC-10 , Issue: 3), 1961; 366 – 370.
[8] Chang S. On the Execution of Fuzzy Programs Using Finite-State Machines. Computers, IEEE Transactions (Volume:C-21 , Issue: 3), 1972; 241 – 253.
[9] Chu T.A, Mani N, Leung C.K.C. A new state assignment technique for asynchronous finite state machines. VLSI, 1993. 'Design Automation of High Performance VLSI Systems', Proceedings, Third Great Lakes Symposium, 1993; 139 – 143.
[10] Chen Y, Lin F. Modeling of discrete event systems using finite state machines with parameters. Control Applications, 2000. Proceedings of the 2000 IEEE International Conference, 2000; 941 – 946.
[11] Kalay U, Venkataramaiah N, Mishchenko A, Hall D.V, Perkowski M.A. Highly testable finite state machines based on EXOR logic. Communications, Computers and Signal Processing, 1999 IEEE Pacific Rim Conference, 1999; 440 – 443.
[12] Rietsche G. State assignment for finite state machines using T flip-flops. Design Automation Conference, 1993, with EURO-VHDL '93. Proceedings EURO-DAC '93, 1993;396 – 401.