Lab Manual for Embedded System Design

Lab No. 5

Finite State Machine Programming in Arduino

Objectives

In this lab students are introduced with the concept of Finite State Machine and its Programming in Arduino. Some application designs will be implemented using FSM Programming

LAB # 5
Finite State Machine Programming in Arduino

Introduction

Finite State Machine:

A finite-state machine (FSM) or finite-state automaton (FSA), or simply a state machine, is a mathematical model of computation. It is an abstract machine that can be in exactly one of a finite number of states at any given time. The FSM can change from one state to another in response to some external inputs; the change from one state to another is called a transition. A FSM is defined by a list of its states, its initial state, and the conditions for each transition. The behavior of state machines can be observed in many devices in modern society that perform a predetermined sequence of actions depending on a sequence of events with which they are presented. Examples are vending machines, which dispense products when the proper combination of coins is deposited, elevators, whose sequence of stops is determined by the floors requested by riders, traffic lights, which change sequence when cars are waiting, and combination locks, which require the input of combination numbers in the proper order. The finite state machine has less computational power than some other models of computation such as the Turing machine. The computational power distinction means there are computational tasks that a Turing machine can do but a FSM cannot. This is because the number of states limits a FSM's memory it has. FSMs are studied in the more general field of automata theory.

Time Boxing

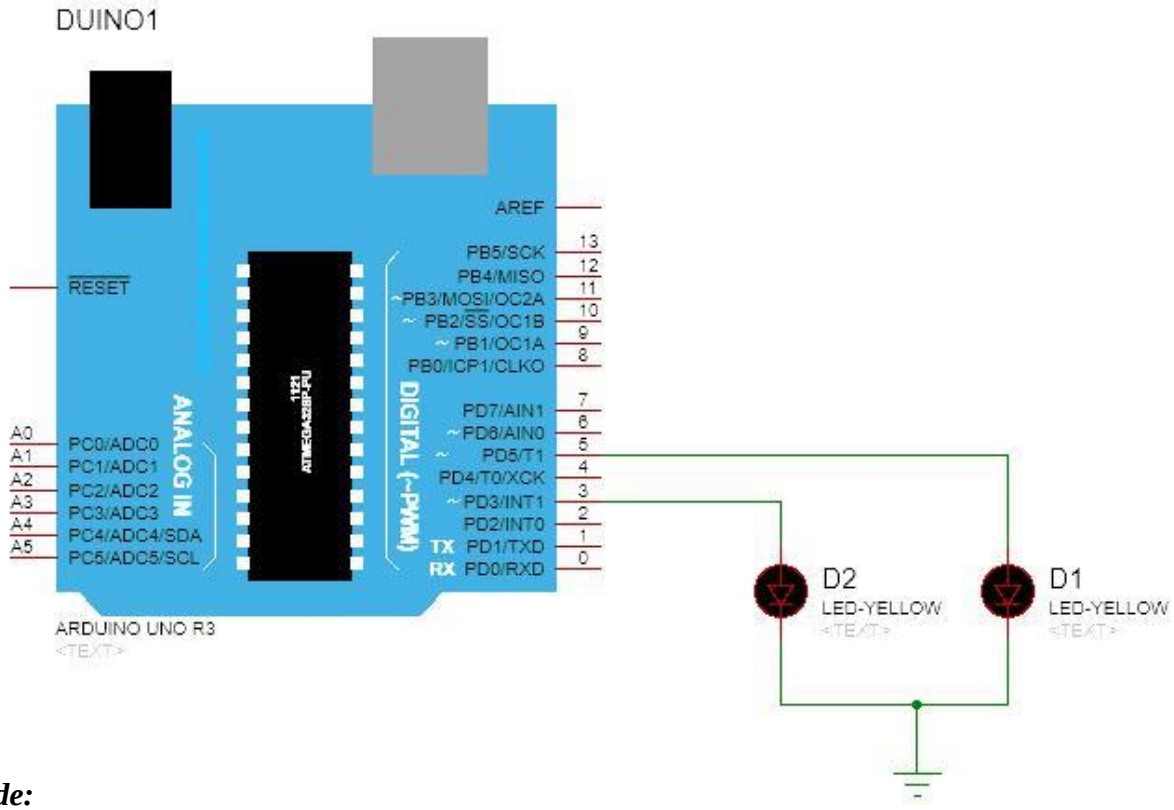
Activity Name	Activity Time	Total Time
Login Systems + Setting up Proteus & Arduino Environment	3 mints + 5 mints	8 mints
Walk through Theory & Tasks	60 mints	60 mints
Implement Tasks	80 mints	80 mints
Evaluation Time	30 mints	30 mints
	Total Duration	178 mints

Objectives

In this lab students are introduced with the concept of Finite State Machine and its Programming in Arduino. Some application designs will be implemented using FSM Programming.

(Arduino State Management):

1. **Write a sketch to blink the 2 LEDs interfaced with Arduino at a different rate simultaneously. (i.e. *"delay"* function limits the designer to perform multitasking from the controller, so this sketch is implement without utilizing this function).**



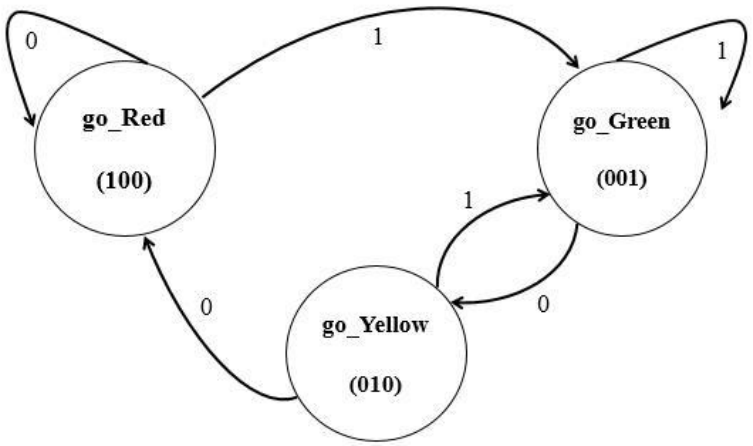
Code:

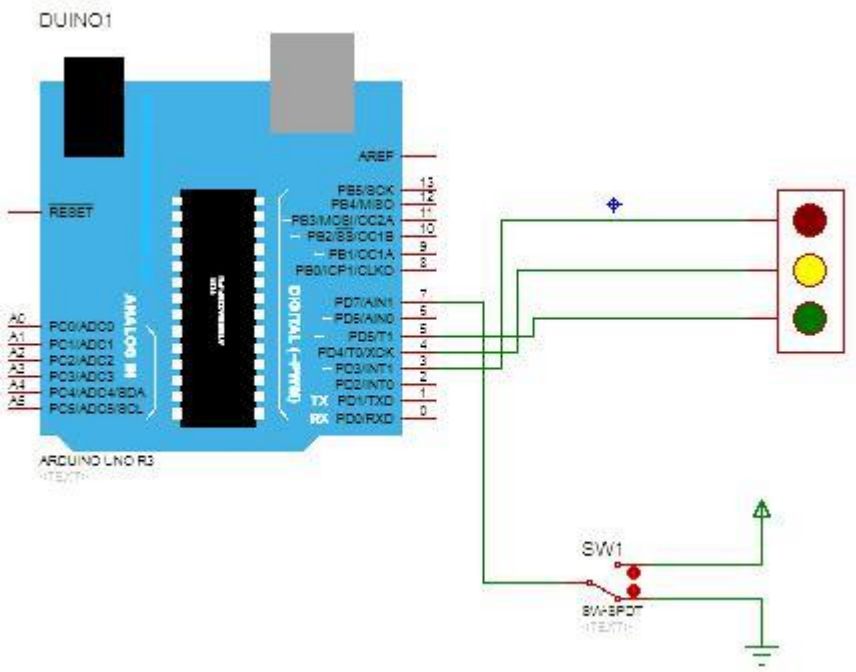
```
const int led1 = 3;
const int led2 = 5;

int led1state = HIGH;
int led2state = HIGH;

long previoustimeled1 = 0;
long previoustimeled2 = 0;

long led1interval = 1000;
long led2interval = 400;

void setup()
{
  pinMode(led1, OUTPUT);

  pinMode(led2, OUTPUT);
}

void loop()
{
  unsigned long currenttime = millis();
```

```
  if (currenttime-previoustimeled1 > led1interval)
  {
    toggleled1();
    previoustimeled1=currenttime;
  }

  if (currenttime-previoustimeled2 > led2interval)
  {
    toggleled2();
    previoustimeled2=currenttime;
  }
}

void toggleled1()
{
  led1state = (led1state == HIGH) ? LOW : HIGH;
  digitalWrite (led1, led1state);
}

void toggleled2()
{
  led2state = (led2state == HIGH) ? LOW : HIGH;
  digitalWrite (led2, led2state);
}
```

(Finite State Machine Design):

2. **Write a sketch to implement the one-way traffic light controller using FSM concepts. The sensor will work to sense the traffic on the road whose output will be the stimulus for the state transition.**





Code:

```
#define goRed 0
#define goYellow 1
#define goGreen 2

int red = 3;
int yellow = 4;
int green = 5;

int sen = 7;

int senState;
static unsigned long ts; int state=goRed; bool flag=HIGH;

void setup()
{
  pinMode(red, OUTPUT); pinMode(yellow, OUTPUT); pinMode(green, OUTPUT);

  pinMode(sen, INPUT);
}

void loop()
{
   senState=digitalRead(sen);

  switch(senState)
  {
```

```
      case HIGH:
      switch (state)
      {
        case goGreen: digitalWrite(red,LOW); digitalWrite(yellow,LOW);
        digitalWrite(green,HIGH); state=goGreen; break;

     case goYellow: digitalWrite(red,LOW); digitalWrite(yellow,HIGH);   // When Sensor is Sensing the
     Traffic.
        digitalWrite(green,LOW); state=goGreen; break;

        case goRed:
        digitalWrite(red,HIGH);
        digitalWrite(yellow,LOW);
        digitalWrite(green,LOW);
        state=goGreen;
        break;
      }
      break;

      case LOW:
      switch (state)
      {
        case goGreen: digitalWrite(red,LOW); digitalWrite(yellow,LOW);
        digitalWrite(green,HIGH); state=goYellow; break;

        case goYellow:
        if(flag){
        ts=millis();
        }
        flag=LOW;
digitalWrite(red,LOW);                          // When Sensor is Not Sensing the Traffic.

        digitalWrite(yellow,HIGH);
        digitalWrite(green,LOW);
        if (millis()>ts+2000)
        {
          state=goRed;
          flag=HIGH;
        }
        break;

        case goRed:
        digitalWrite(red,HIGH);
        digitalWrite(yellow,LOW);
        digitalWrite(green,LOW);
        state=goRed;
        break;
      }
      break;
    }

  }
```