FSM (Finite State Machine) Project

Similar to lab 8 in your lab documentation, you are to implement a FSM to emulate the behavior of a vending machine for dispensing 15 cent gum packs in response to users inserting nickels or dimes into the machine. The rising edge of BTN0 will be used to simulate the insertion of a nickel (N), the rising edge of BTN1 simulates the insertion of a dime (D), and the reflective light sensor (RLS) will act as the Cancel button when the user's finger is placed over this sensor. When the microcontroller code is initiated, the vending machine will have 7 packs of gum in it from the start. The FSM should execute every 0.2 seconds.

The states of the machine would be the following:

1. Ready — Machine is ready to accept cash. LED0 is off and LED1 is off to indicate this state.
2. CashCollected — Once the machine has collected the correct cash (15 cents), the user can obtain the gum or cancel the transaction if less than 15 cents has been placed in the machine. LED0 is on and LED1 is off to indicate this state.
3. DispenseChange — Give back the change to the user if more than 15 cents (2 dimes) is placed in the machine. LED0 is off and LED1 is on to indicate this state.
4. DispenseItem — Dispense the gum upon the collection of 15 cents. LED0 is on and LED1 is on to indicate this state.
5. TransactionCancelled — If the user cancels the transaction, return the cash given by the user. The buzzer generates a tone of a certain frequency (your choice) to denote that the cancellation state has occurred.
6. LowInventoryWarning — If the inventory falls below 5 gum packs, then warn the distributor of low inventory. Send emails to the "distributor" every time the count is below 5. The buzzer generates a different tone from the tone used in the Cancel state to indicate the low inventory state each time a pack of gum is dispensed.
7. NoInventory — If the inventory becomes zero, then have LED0 and LED1 flash at different rates, no coins can be accepted, and the FSM stays in this state. The only way to get out of this state is to replenish the gum packs by pressing the tiny reset button on the Particle Photon board to re-initialize the machine and start in the Ready state.

The following is example code of an FSM controlling a two bit up/down counter. Use this code as a template in structuring your vending machine FSM.

```
// two-bit up/down counter FSM: B0 = count down, B1 = count up

volatile int B0, B1, tick; // button and timer flags

void C0(void) { tick = 1; } // FSM clock

Timer t0( 1000, C0); // 1 second

void setup() {
  pinMode( D0, OUTPUT); pinMode( D7, OUTPUT); // LEDs
  pinMode( A0, INPUT_PULLDOWN); pinMode( A1, INPUT_PULLDOWN); // buttons
  t0.start(); // timer
}

enum states { s00, s01, s10, s11 }; // enum creates distinct integer constants
```

```
enum states state = s00, next; // current and next state variables

void FSM(void) {
  B0 = digitalRead( A0); B1 = digitalRead( A1); // inputs
  next = state; // stay in current state by default
  switch( state) {
    case s00: digitalWrite( D0, 0); digitalWrite( D7, 0);
      if( B0) next = s11; else if( B1) next = s01;
      break;
    case s01: digitalWrite( D0, 0); digitalWrite( D7, 1);
      if( B0) next = s00; else if( B1) next = s10;
      break;
    case s10: digitalWrite( D0, 1); digitalWrite( D7, 0);
      if( B0) next = s01; else if( B1) next = s11;
      break;
    case s11: digitalWrite( D0, 1); digitalWrite( D7, 1);
      if( B0) next = s10; else if( B1) next = s00;
      break;
  }
  state = next; // update state
}

void loop() {
  if(tick) { FSM(); tick = 0; }
}
```

During lab, demonstrate your program to your instructor or the TA. Also upload your cpp file to Black Board and make sure that your code is well documented with comments.