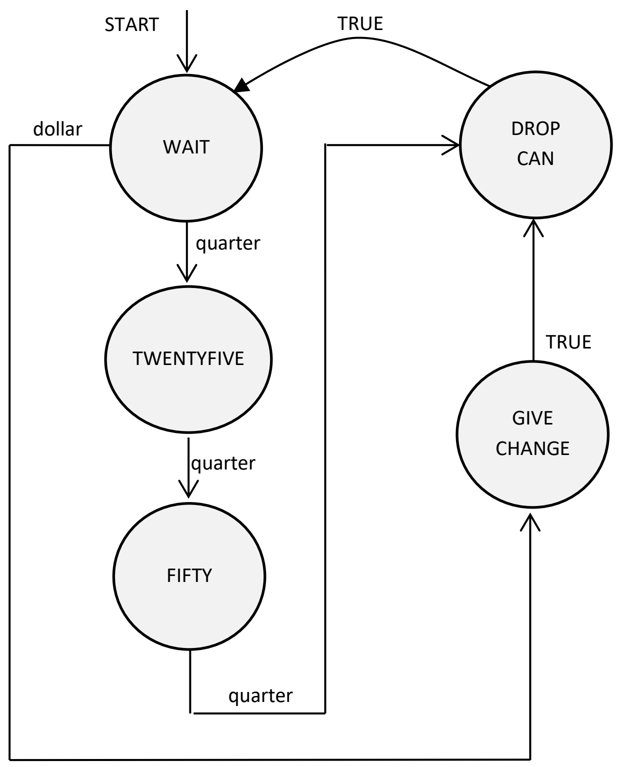
State machine with Arduino (Part 2)

The vending machine

In this part, we will simulate a simple vending machine. To make matters simple, we will agree that:

- This machine distributes bottles of water.
- Each bottle sells for 75¢.
- Only dollars and quarters are accepted.
- When a dollar is used to pay, a quarter will be returned for change.
- When a quarter has been inserted, dollars are not accepted anymore.

We will use 2 buttons: One to simulate dropping a quarter and another one to simulate dropping a dollar.
We will use two LEDs: One to simulate that the can has been dropped, and another one to simulate that the change has been given.



This machine will have 5 states:

WAIT :	Probe the two switches: If dollar: change state to GIVE CHANGE state If quarter: change state to TWENTYFIVE state
TWENTYFIVE:	Probe the quarter switch If quarter: change state to FIFTY state
FIFTY:	Probe the quarter switch If quarter: change state to DROP CAN state
GIVE CHANGE:	Flash LED Change state to DROP CAN state
DROP CAN:	Flash LED Change state to WAIT state

First, we define the state machine:

```
enum VendingMachineStates {WAIT, TWENTYFIVE, FIFTY, GIVE_CHANGE, DROP_CAN};
VendingMachineStates vmState = WAIT;

void vendingMachine() {  //We will complete it later
  switch (vmState) {
    case WAIT: { break: }
    case TWENTYFIVE { break: }
    case FIFTY: { break: }
    case GIVE_CHANGE: { break: }
    case DROP_CAN: { break: }
  }
}
```

We will use the debouncer:

```
#include <EdgeDebounceLite.h>
EdgeDebounceLite debounce;
```

We will need to read two switches. We will use the switch state machine that we created in Part 1. We will just modify it so that it can use more than one switch (changes are in blue).

```
enum SwitchStates {IS_OPEN, IS_RISING, IS_CLOSED, IS_FALLING};
SwitchStates switchState[2] = {IS_OPEN, IS_OPEN};
byte switchPin[2] = {10, 11};

enum SwitchModes {PULLUP, PULLDOWN};
SwitchModes switchMode[2] = {PULLUP, PULLUP};

void switchMachine(byte i) {
  byte pinIs = debounce.pin(switchPin[i]);
  if (switchMode[i] == PULLUP) pinIs = !pinIs;
  switch (switchState[i]) {
    case IS_OPEN:    { if(pinIs == HIGH) switchState[i] = IS_RISING;  break; }
    case IS_RISING:  {                   switchState[i] = IS_CLOSED;  break; }
    case IS_CLOSED:  { if(pinIs == LOW)  switchState[i] = IS_FALLING; break; }
    case IS_FALLING: {                   switchState[i] = IS_OPEN;    break; }
  }
}
```

We will create two functions to read the switches:

```
bool dollarInserted() {
  switchMachine(0);
  if (switchState[0] == IS_FALLING) return true;
  else                              return false;
}

bool quarterInserted() {
  switchMachine(1);
  if (switchState[1] == IS_FALLING) return true;
  else                              return false;
}
```

We will just quickly blink the LEDs so we can afford to use delay() for this.

```
byte ledPin[2] = {2, 3};

void blinkLed(byte i) {
  digitalWrite(ledPin[i], HIGH);
  delay(10);
  digitalWrite(ledPin[i], LOW);
}
```

We will create two functions to blink the LEDs

```
void giveChange() {
  blinkLed(0);
}

void dropCan() {
  blinkLed(1);
}
```

Now we can complete the vending machine code.

```
void vendingMachine() {
  switch (vmState) {
    case WAIT:        { if (dollarInserted())  vmState = GIVE_CHANGE;
                        if (quarterInserted()) vmState = TWENTYFIVE;   break: }
    case TWENTYFIVE   { if (quarterInserted()) vmState = FIFTY;        break: }
    case FIFTY:       { if (quarterInserted()) vmState = DROP_CAN;     break: }
    case GIVE_CHANGE: { giveChange();          vmState = DROP_CAN;     break: }
    case DROP_CAN:    { dropCan();             vmState = WAIT;         break: }
  }
}
```

Putting it all together with debounced switches gives:

```
#include <EdgeDebounceLite.h>
EdgeDebounceLite debounce;

enum VendingMachineStates {WAIT, TWENTYFIVE, FIFTY, GIVE_CHANGE, DROP_CAN};
VendingMachineStates vmState = WAIT;

enum SwitchStates {IS_OPEN, IS_RISING, IS_CLOSED, IS_FALLING};
SwitchStates switchState[2] = {IS_OPEN, IS_OPEN};
byte switchPin[2] = {10, 11};

enum SwitchModes {PULLUP, PULLDOWN};
SwitchModes switchMode[2] = {PULLUP, PULLUP};

byte ledPin[2] = {2, 3};

void switchMachine(byte i) {
  byte pinIs = debounce.pin(switchPin[i]);
  if (switchMode[i] == PULLUP) pinIs = !pinIs;
  switch (switchState[i]) {
    case IS_OPEN:    { if(pinIs == HIGH) switchState[i] = IS_RISING;  break; }
    case IS_RISING:  {                   switchState[i] = IS_CLOSED;  break; }
    case IS_CLOSED:  { if(pinIs == LOW)  switchState[i] = IS_FALLING; break; }
    case IS_FALLING: {                   switchState[i] = IS_OPEN;    break; }
  }
}


bool dollarInserted() {
  switchMachine(0);
  if (switchState[0] == IS_FALLING) return true;
  else                              return false;
}


bool quarterInserted() {
  switchMachine(1);
  if (switchState[1] == IS_FALLING) return true;
  else                              return false;
}
void blinkLed(byte i) {
  digitalWrite(ledPin[i], HIGH);
  delay(10);
  digitalWrite(ledPin[i], LOW);
}
```

```
void giveChange() {
  blinkLed(0);
}

void dropCan() {
  blinkLed(1);
}
void vendingMachine() {
  switch (vmState) {
    case WAIT:        { if (dollarInserted())  vmState = GIVE_CHANGE;
                        if (quarterInserted()) vmState = TWENTYFIVE;   break; }
    case TWENTYFIVE:  { if (quarterInserted()) vmState = FIFTY;        break; }
    case FIFTY:       { if (quarterInserted()) vmState = DROP_CAN;     break; }
    case GIVE_CHANGE: { giveChange();          vmState = DROP_CAN;     break; }
    case DROP_CAN:    { dropCan();             vmState = WAIT;         break; }
  }
}

void setup() {
  for (int i = 0 ; i < 2 ; i++) pinMode(switchPin[i], INPUT_PULLUP);
  for (int i = 0 ; i < 2 ; i++) pinMode(ledPin[i], OUTPUT);
}
void loop() {
  vendingMachine();
}
```

In the next part, we will simulate an elevator.