State machine with Arduino (Part 2)

The vending machine

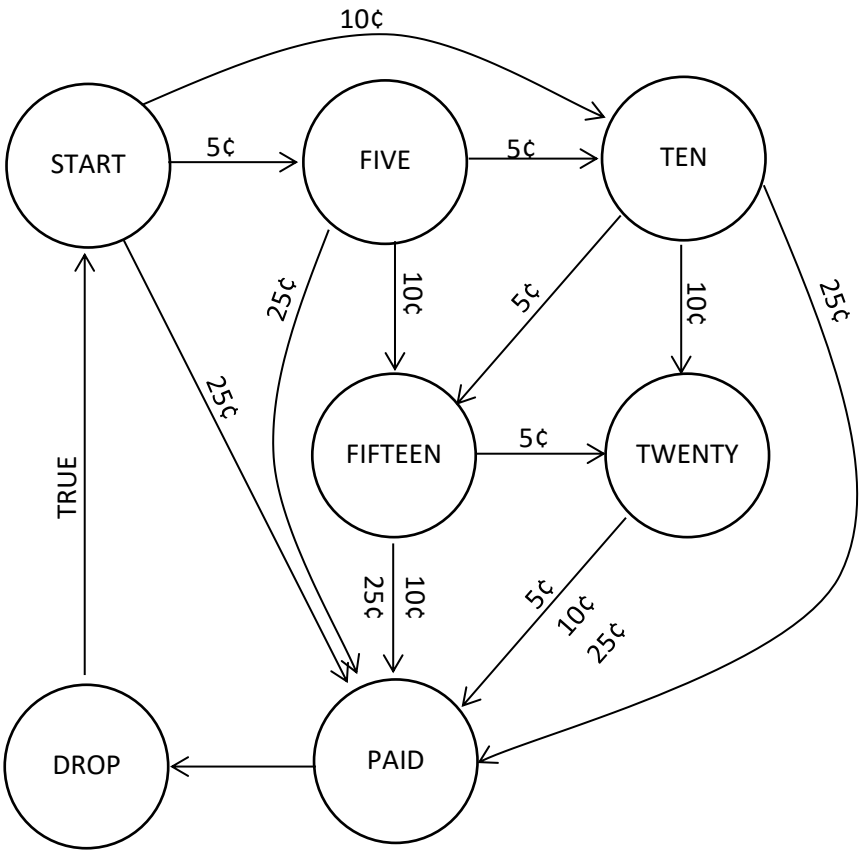
In this part, we will simulate a simple vending machine. To make matters simple, we will agree that:

- there is only one product that can be bought from this machine;
- this product costs 25¢;
- this machine accepts nickels, dimes and quarters;
- this machine does not give back change.

To be able to simulate it without having to actually build the machine we will need:

- one button that simulates that a nickel has been dropped in the machine;
- one button that simulates that a dime has been dropped in the machine;
- one button that simulates that a quarter has been dropped in the machine;
- a LED that lights up to simulate that the product has been dropped in the bin.

The state machine's schematic looks like this: The nodes represent the total amount of money that has been dropped in the machine. The transitions say that a piece of change (5¢, 10¢ or 25¢) has been dropped.



We will use our switch state machine from Part 1 to read the switches:

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

enum SwitchStates {OPENs, RISINGs, CLOSEDs, FALLINGs[1]};
enum ButtonTypes {PULLUP, PULLDOWN};
SwitchStates sState[3] = {OPENs, OPENs, OPENs};
ButtonTypes buttonType[3] = {PULLUP, PULLUP, PULLUP};
byte buttonPins[3] = {4, 5, 6};

void setup() {
  for (byte i = 0 ; i < 3 ; i++) pinMode(buttonPins[i], INPUT_PULLUP);
}

void readSwitch(byte i) {
  byte pinStatus = debounce.pin(buttonPins[i]);
  if (buttonType[i] == PULLUP) pinStatus = !pinStatus;
  switch (sState[i]) {
    case OPENs:    { if (pinStatus == HIGH} sState[i] = RISINGs;  break; }
    case RISINGs:  {                        sState[i] = CLOSEDs;  break; }
    case CLOSEDs:  { if (pinStatus == LOW)  sState[i] = FALLINGs; break; }
    case FALLINGs: {                        sState[i] = OPENs;    break; }
  }
}
```

We need functions to read the buttons to be able to change the vending state machine's states:

```
bool nickelDropped() {
  readSwitch(0);
  if (sState[0] == FALLINGs) return true;
  else                       return false;
}
bool dimeDropped() {
  readSwitch(1);
  if (sState[1] == FALLINGs) return true;
  else                       return false;
}
bool quarterDropped() {
  readSwitch(2);
  if (sState[2] == FALLINGs) return true;
  else                       return false;
}
```

We need to drop the product:

```
byte ledPin = 13

void setup() {
  pinMode(ledPin, OUTPUT);
}

void dropProduct() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
}
```

[1] Had to add an 's' at the end of the states because Arduino has already #defined RISING

Now, we will translate our schematic into Arduino code.

```
enum VendingMachineStates {START, FIVE, TEN, FIFTEEN, TWENTY, PAID, DROP};
VendingMachineStates vmState = START;

void vendingMachine() {
  switch (vmState) {
    case(START):   { if (quarterDropped()) vmState = PAID;
                     if (dimeDropped())    vmState = TEN;
                     if (nickelDropped())  vmState = FIVE;    break; }
    case(FIVE):    { if (quarterDropped()) vmState = PAID;
                     if (dimeDropped())    vmState = FIFTEEN;
                     if (nickelDropped())  vmState = TEN;     break; }
    case(TEN):     { if (quarterDropped()) vmState = PAID;
                     if (dimeDropped())    vmState = TWENTY;
                     if (nickelDropped())  vmState = FIFTEEN; break; }
    case(FIFTEEN): { if (quarterDropped()) vmState = PAID;
                     if (dimeDropped())    vmState = PAID;
                     if (nickelDropped())  vmState = TWENTY;  break; }
    case(TWENTY):  { if (quarterDropped()) vmState = PAID;
                     if (dimeDropped())    vmState = PAID;
                     if (nickelDropped())  vmState = PAID;    break; }
    case(PAID):    { dropProduct();        vmState = START;   break; }
  }
}
```

And here is the complete code:

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

enum SwitchStates {OPENs, RISINGs, CLOSEDs, FALLINGs};
SwitchStates sState[3] = {OPENs, OPENs, OPENs};
enum ButtonTypes {PULLUP, PULLDOWN};
ButtonTypes buttonType[3] = {PULLUP, PULLUP, PULLUP};
byte buttonPins[3] = {4, 5, 6};
byte ledPin = 13;
enum VendingMachineStates {START, FIVE, TEN, FIFTEEN, TWENTY, PAID, DROP};
VendingMachineStates vmState = START;

void readSwitch(byte i) {
  byte pinStatus = debounce.pin(buttonPins[i]);
  if (buttonType[i] == PULLUP) pinStatus = !pinStatus;
  switch (sState[i]) {
    case OPENs:    { if (pinStatus == HIGH) sState[i] = RISINGs;  break; }
    case RISINGs:  {                        sState[i] = CLOSEDs;  break; }
    case CLOSEDs:  { if (pinStatus == LOW)  sState[i] = FALLINGs; break; }
    case FALLINGs: {                        sState[i] = OPENs;    break; }
  }
}
bool nickelDropped() {
  readSwitch(0);
  if (sState[0] == FALLING) return true; else return false;
}
bool dimeDropped() {
  readSwitch(1);
  if (sState[1] == FALLING) return true; else return false;
}
bool quarterDropped() {
  readSwitch(2);
  if (sState[2] == FALLING) return true; else return false;
}
```

```
void dropProduct() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
}

void vendingMachine() {
  switch (vmState) {
    case(START):   { if (quarterDropped()) vmState = PAID;
                     if (dimeDropped())    vmState = TEN;
                     if (nickelDropped())  vmState = FIVE;    break; }
    case(FIVE):    { if (quarterDropped()) vmState = PAID;
                     if (dimeDropped())    vmState = FIFTEEN;
                     if (nickelDropped())  vmState = TEN;     break; }
    case(TEN):     { if (quarterDropped()) vmState = PAID;
                     if (dimeDropped())    vmState = TWENTY;
                     if (nickelDropped())  vmState = FIFTEEN; break; }
    case(FIFTEEN): { if (quarterDropped()) vmState = PAID;
                     if (dimeDropped())    vmState = PAID;
                     if (nickelDropped())  vmState = TWENTY;  break; }
    case(TWENTY):  { if (quarterDropped()) vmState = PAID;
                     if (dimeDropped())    vmState = PAID;
                     if (nickelDropped())  vmState = PAID;    break; }
    case(PAID):    { dropProduct();        vmState = START;   break; }
  }
}
void setup() {
  for (byte i = 0 ; i < 3 ; i++) pinMode(buttonPins[i], INPUT_PULLUP);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  vendingMachine();
}
```