LAB1: Operating MiniFan Using Mealy Machine

Date: 2023.09.13

Author/Partne: Noh Yunki / Kim EunChan

Demo Video: https://youtu.be/1SxNmhlXJX0

1. Introduction

In this lab, we set the goal to operate minifan using structure of Mealy Machine. To operate minifan, STM board recieves button input and distance input. As button is pressed, the state of minifan will be changed and depending on whether the distance from Ultrasonic sensor is under threshold value or not, the minifan will work or paused. These functions of minifan have to be formed in Mealy Machine process. The imformation about Mealy Machine and table of minifan's state and output are provided at the '**3. Problem**' stage.

2. Rquirement

Hardware

- MCU
- NUCLEO-F401RE
- Sensor :
- - Ultrasonic distance sensor(HC-SR04)
  - Button
- Actuator / Display
- - LED
  - DC Motor

Software

- Arduino IDE

3. Problem

Procedure

1. DC Motor Output

- DC Motor output types are three types(speed: 0, 255/2, 255). These types are change when button is pressed. In addition to this, distance information from Ultrasonic sensor also changes DC Motor's

output. When distance value is under 5[cm], DC Motor will not be paused. However in the case of over 5cm, DC Motor will be paused even when DC Motor's state is not 0 speed.

2. LED Output

- LED will blink when button is pressed as a signal of output.

4. Configuration

Pins information

DC Motor

- Pin: D11

Button

- Pin: 3

LED

- Pin: D13

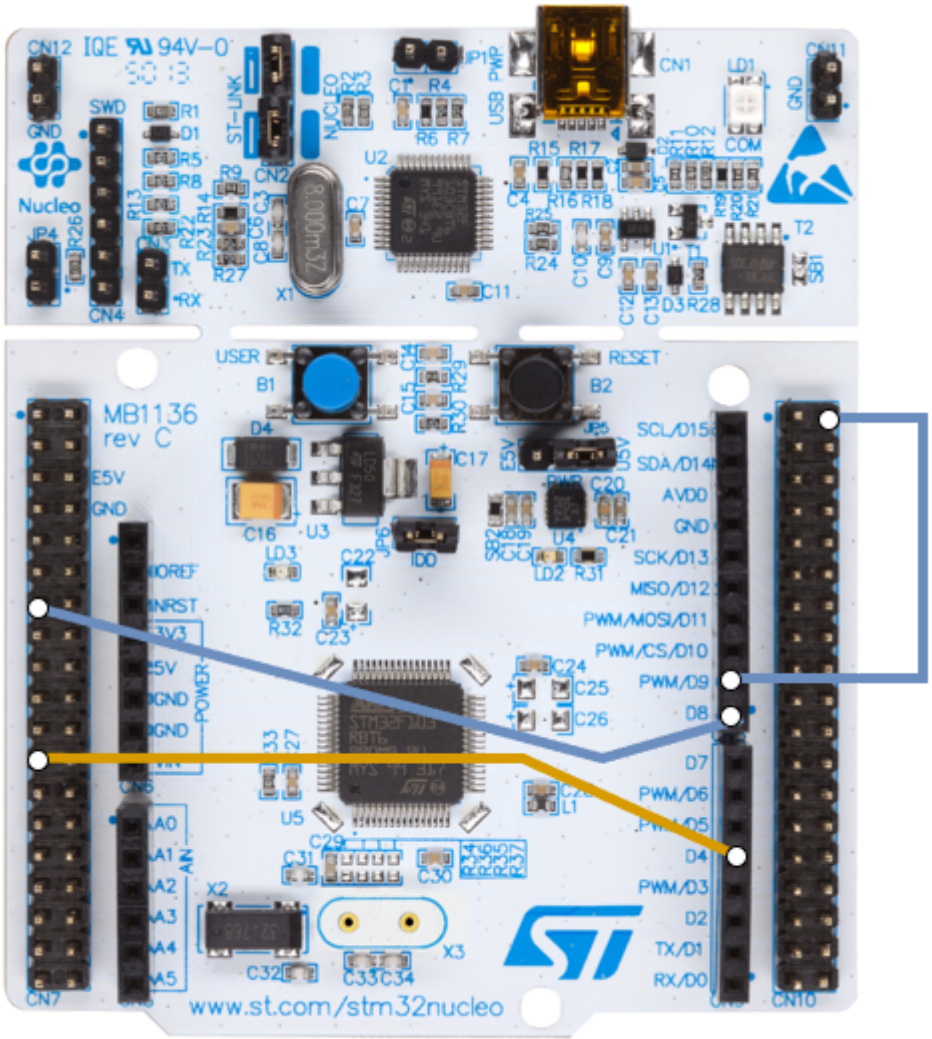
Trigger

- Pin: D10

Echo

- Pin: D7

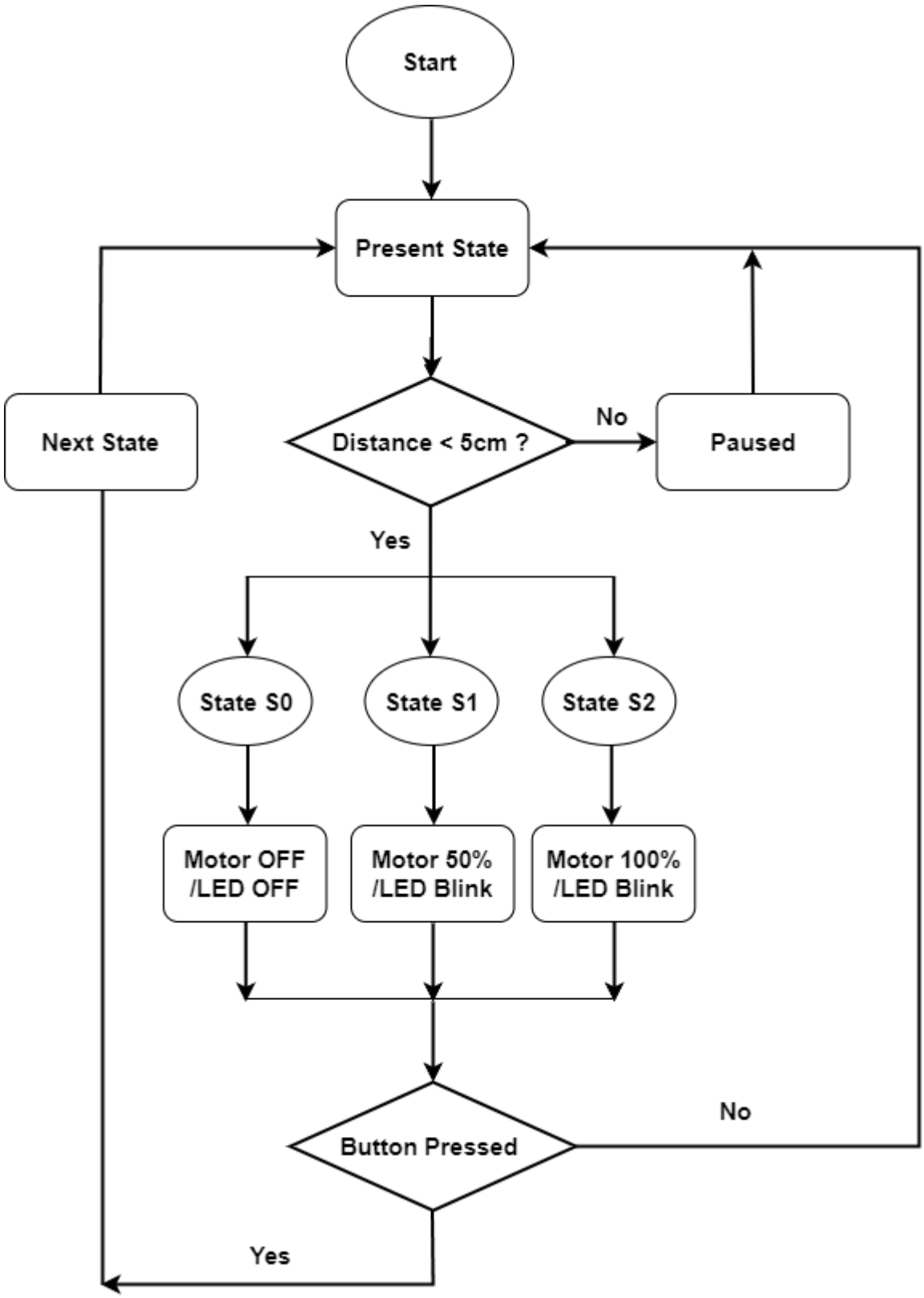
Circuit



- Image 1. stm32nucleo Circuit

5. Algorithm

Flow chart



- Flow chart 1. Mealy Machine's Smart Fan Algorithm

I/O Table

Present State	Next State (X, Y)				Output Z			
	(O, F)	(O, T)	(1, F)	(1, T)	(0, F)	(0, T)	(1, F)	(1, T)
S0	S0	S0	S1	S1	V=0 L=0	V=0 L=0	V=0 L=0	V=50 L=1
S1	S1	S1	S2	S2	V=0 L=1	V=50 L=1	V=50 L=1	V=100 L=1
S2	S2	S2	S0	S0	V=0 L=1	V=100 L=1	V=100 L=1	V=0 L=0

- Table 1. Input/Output Table

There are two inputs(Pressing Button, Distance value). And these inputs have different impacts on the Output(DC Motor, LED). The state of system is changed only when Button is pressed. So, when the Button is pressed in order, the DC Motor's speed will be changed like this, "S0: 0% speed --> S1:50% speed --> S2: 100% speed". In addition to this, the LED will blink except S0 state. In this reason, Button also has impact on LED Output. About the imapct of Distance value, this value has impact of pausing the DC Motor. Because the threshold value of Distance is 5[cm], when Supersonic distance sensor's distance value is under 5[cm], the DC Motor will work as the FSM state works. However when the distance value is over 5[cm], the DC Motor will be paused.

However because we are trying to operate these operations in Mealy Machine mechanism, it is important to printing out DC Motor's output of present state before change present state with next state. In this reason, I set the "Button pressed" box at the end position.

Decription with Code

- Lab source code

```
/*Mealy: 입력을 하면 반응(출력)이 먼저 나오고 상태(next state)가 갱신된다
  Moore: 입력을 하면 상태(next state)가 먼저 갱신되고 반응(출력)이 뒤따라 온다.

  그렇기에  Mealy는 현 상태에서의 입력에 따라 바로 따라오는 출력을 찾아야 하고,
            Moore는 다음 상태를 먼저 갱신하고 이 상태에서의 출력값을 찾아야 한다.

  그래서 Moore에서는

    void stateOutput(){
      pwmOut = FSM[state].out[PWM];
      ledOut = FSM[state].out[LED];

   이 코드에서 state에 해당하는 값이 next state이고 이에 따른 결과를 PWM, LED라는
   지정된 위치에서 뽑아내는 것이다.
}
*/

// State definition
#define S0  0
#define S1  1
#define S2  2
```

```
// Pin setting
const int ledPin = 13;
const int pwmPin = 11;
const int btnPin = 3;
const int trigPin = 10;
const int echoPin = 7;

// Initialize
unsigned char state = S0;
unsigned char input[2] = {0, 0}; // input[0] = button, input[1] = distance
unsigned char pwmOut = 0;
unsigned char ledOut = LOW;
unsigned long duration;

float distance;
int thresh = 5;

float time = 0;

// State table definition
typedef struct {
  uint32_t out[2][2];      // output = FSM[state].out[PWM or LED]
  uint32_t next[2][2];  // nextstate = FSM[state].next[input X][input Y]
} State_t;

// state[three rows]: 3 types[S0, S1, S2],
// next state[first column]: 4 types per each state,
// output of pwm[second column]: 4 reuslts per each type occured from led and distance
inputs
State_t FSM[3] = {
  { {{0  , 0 }, {0, 255/2}}, {{S0 , S0}, {S1 , S1}} },
  { {{0  , 255/2 }, {0, 255}}, {{S1 , S1}, {S2 , S2}} },
  { {{0  , 255 }, {0, 0}}, {{S2 , S2}, {S0 , S0}} },
};

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);

  // Initialize pwm pin as an output:
  pinMode(pwmPin, OUTPUT);

  // initialize the pushbutton pin as an interrupt input:
  pinMode(btnPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(btnPin), pressed, FALLING);

  // Initialize the trigger pin as an output
  pinMode(trigPin, OUTPUT);

  // Initialize the echo pin as an input
  pinMode(echoPin, INPUT);


  Serial.begin(9600);
```

```
}

void loop() {
   // Generate pwm singal on the trigger pin.
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  delayMicroseconds(10);

  // Distance is calculated using how much time it takes.
  duration = pulseIn(echoPin, HIGH);
  distance = (float)duration / 58.0;

  // Output State
  stateOutput();

  analogWrite(pwmPin, pwmOut);
  digitalWrite(ledPin, ledOut);

  // Calculate next state, then update State
  nextState();

  // Print
  Serial.print("Input = ");
  Serial.print(input[1]);

  Serial.print("    State = ");
  Serial.print(state);

  Serial.print("    distance = ");
  Serial.print(distance);
  Serial.println(" [cm]");

  Serial.print("Output : ");
  Serial.print("LED = ");
  if (state == S0){
    Serial.print("OFF ");
    }
  else
    Serial.print("Blink ");

  Serial.print("    PWM = ");
  Serial.println(pwmOut);

  Serial.println(" ");


  delay(1000);
}


void pressed(){
```

```
  input[0] = 1;
  // print out Output before updating present state with next state
  stateOutput();
  nextState();
  // reset the value of input[0](button pressing)
  input[0] = 0;
}

void nextState(){
  state = FSM[state].next[input[0]][input[1]];
}

void stateOutput(){
  // distance value input
  if (distance < thresh)
    input[1] = 1;
  else
    input[1] = 0;

  // Button pressing input
  if (state == S0){
    ledOut = 0;
  }
    // blink
  else{
    if (millis() - time >= 1000){
      if(ledOut == HIGH)
        ledOut = LOW;
      else
        ledOut = HIGH;

      time = millis();
    }
  }

  // Assign pwmOut value and ledOut value
  pwmOut = FSM[state].out[input[0]][input[1]];
}
```

- Next state x 4 & Output x 4

```
typedef struct {
  uint32_t out[2][2];      // output = FSM[state].out[PWM or LED]
  uint32_t next[2][2];  // nextstate = FSM[state].next[input X][input Y]
} State_t;
```

Because Next state and Output's values are four, I set the out matrix and next matrix as 2 x 2 .

- Input & Output Table

```
State_t FSM[3] = {
  { {{0  , 0 }, {0, 255/2}}, {{S0 , S0}, {S1 , S1}} },
  { {{0  , 255/2 }, {0, 255}}, {{S1 , S1}, {S2 , S2}} },
  { {{0  , 255 }, {0, 0}}, {{S2 , S2}, {S0 , S0}} },
};
```

- Mealy Machine Mechanism(Output from present state --> Updating present state with next state
- - When button is pressed(state of FSM is changed)

```
void pressed(){
  input[0] = 1;
  // print out Output before updating present state with next state
  stateOutput();
  nextState();
  // reset the value of input[0](button pressing)
  input[0] = 0;
}
```

```
void stateOutput(){
  // distance value input
  if (distance < thresh)
    input[1] = 1;
  else
    input[1] = 0;

  // Button pressing input
  if (state == S0){
    ledOut = 0;
  }
    // blink
  else{
    if (millis() - time >= 1000){
      if(ledOut == HIGH)
        ledOut = LOW;
      else
        ledOut = HIGH;

      time = millis();
    }
  }
```

```
void nextState(){
  state = FSM[state].next[input[0]][input[1]];
}
```

- - When button is not pressed(State of FWM is not changed)

```
void loop(){


...
```

```
stateOutput();

  analogWrite(pwmPin, pwmOut);
  digitalWrite(ledPin, ledOut);

  // Calculate next state, then update State
  nextState();

...

}
```

Both case show that Outputs are printed before updating Next state.

6. Results and Analysis

Results

As the state of FSM is changed, the outputs of DC Motor and LED are changed well. At the state of S0, DC Motor is turned off. At the S1, DC Motor runs at 50% speed. And at the S2 state, DC Motor runs 100% speed. Also, when the state is not S0, LED blink. As another input, when distance value from Supersonic sensor is over 5[cm], the DC Motor is paused. Even when the Motor is paused, if button pressed, the state of FSM is changed.

Analysis

Performing these operations will be also possible in other ways. However by organizing several states relataed with specific outputs in I/O Table, the same operations are perfomred simply. I'm not quite used to this method yet, but I could learn Mealy Method is efficient at to for Input and Output mechanism.