LAB: Smart mini-fan with STM32-duino

Name/ID: 한태건 21900793

Date: 2023. 09. 14

Demo Video : https://youtu.be/FbGgnJGKj8k

I. Introduction

In this lab, you are required to create a simple program that uses arduino IDE for implementing a simple embedded digital application. Refer to online arduino references for the full list of APIs.

Hardware

- NUCLEO -F401RE or NUCLEO -F411RE
- Ultrasonic distance sensor(HC-SR04)
- DC motor (RK-280RA)
- LED

Software

- Arduino IDE

II. Problem

Procedure

The program needs to run the Fan only when the distance of an object is within a certain value.

Example: An automatic mini-fan that runs only when the face is near the fan. Otherwise turns off.

- As the button **B1** is pressed, change the fan velocity. The MODE(states) are
  - MODE(state): **OFF(0%), MID(50%), HIGH(100%)**
- When the object(face) is detected about 50 mm away, then it automatically pauses the fan temporarily.
  - Even the fan is temporarily paused, the MODE should be changed whenever the button **B1** is pressed
- When the object(face) is detected within 50mm, then it automatically runs the fan
  - It must run at the speed of the current MODE
- LED(**LED1**): Turned OFF when MODE=OFF. Otherwise, blink the LED with 1 sec period (1s ON, 1s OFF)
- Print the distance and PWM duty ratio in Tera-Term console (every 1 sec).
- Must use Mealy FSM to control the mini-fan
  - Draw a FSM(finite-state-machine) table and state diagram
  - Example Table. See below for example codes

Configuration

Ultrasonic distance sensor

Trigger:

- Generate a trigger pulse as PWM to the sensor
- Pin: **D10** (TIM4 CH1)
- PWM out: 50ms period, 10us pulse-width

Echo:

- Receive echo pulses from the ultrasonic sensor
- Pin: **D7** (Timer1 CH1)
- Input Capture: Input mode
- Measure the distance by calculating pulse-width of the echo pulse.

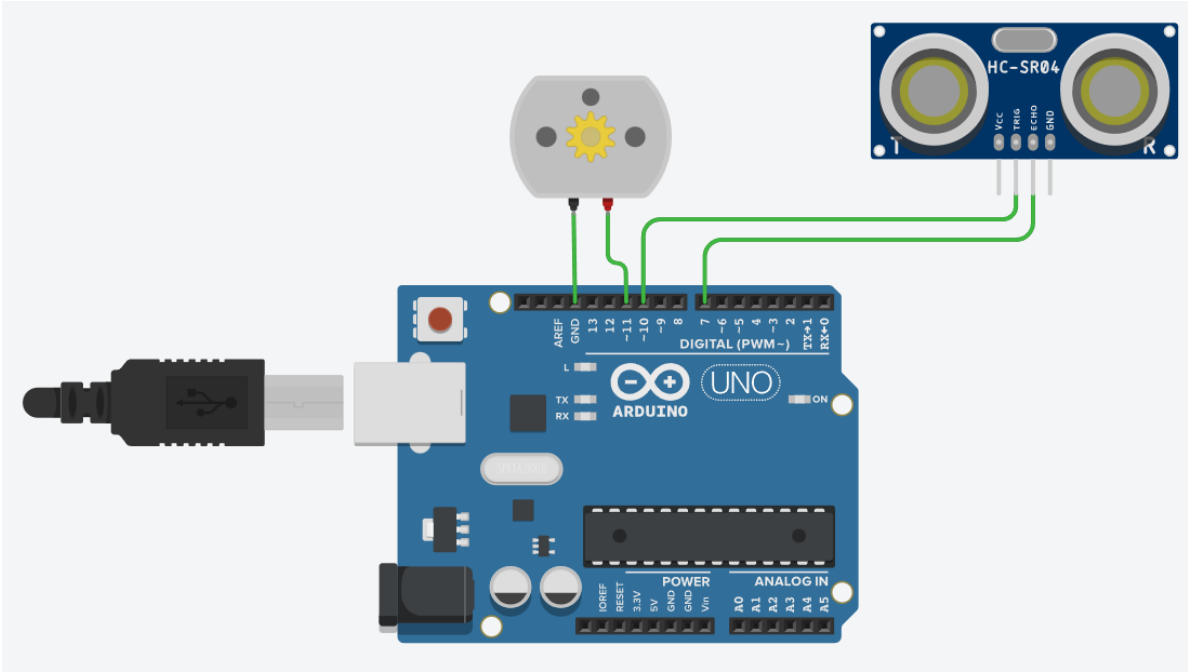
USART

- Display measured distance in [cm] on serial monitor of Tera-Term.
- Baudrate 9600

DC Motor

- PWM: PWM1, set 10ms of period by default
- Pin: **D11** (Timer1 CH1N)

Circuit/Wiring Diagram



III.Algorithm

State Table

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

Table 1. State Table

State Graph

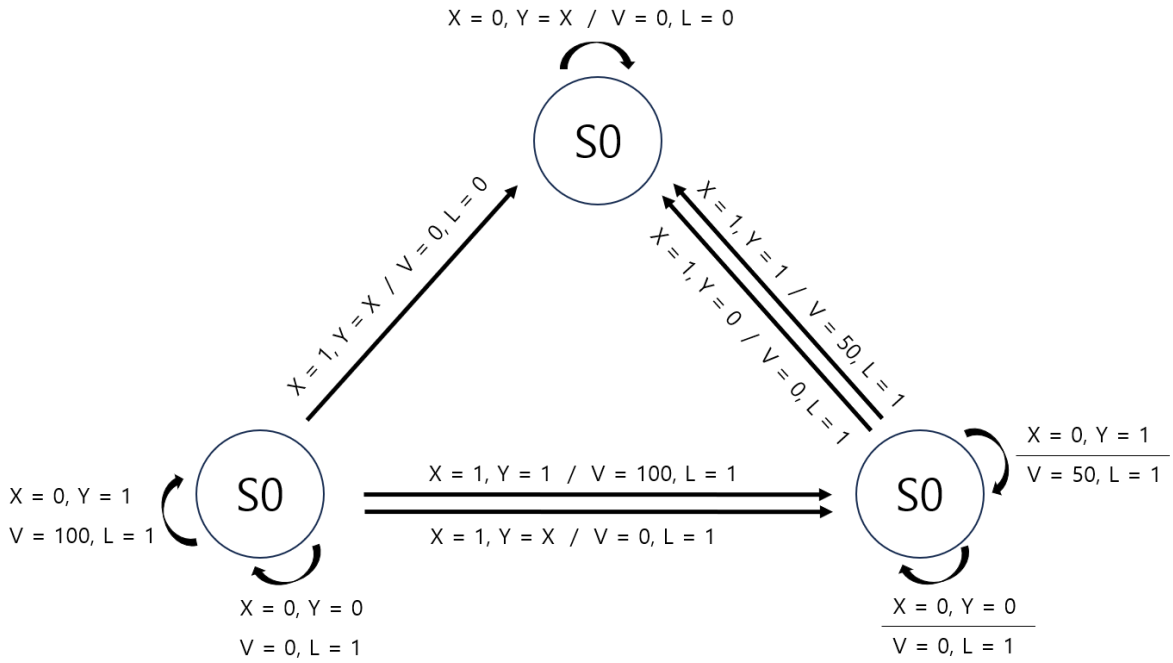


Figure 1. State Graph

Source code

Below are the Arduino code for Smart mini-fan with STM32-duino

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

// Address number of output in array
#define PWM 0
```

```
#define LED 1

// Define Variables
const int ledPin = 13;
const int pwmPin = 11;
const int btnPin = 3;
const int trigPin = 10;   // Trigger pin : PWM out
const int echoPin = 7;    // Echo pin : Interrupt in

unsigned char state = S0;
unsigned char nextstate = S0;
int bPressed = 0;
int ledOn = LOW;

unsigned int input[2] = {0, 0};
unsigned int output[3] = {0, 0, 1};
unsigned char pwmOut = 0;
unsigned long duration;
float distance;
int thresh = 5;
unsigned long time = 0;
```

```
// Define structure
typedef struct {
  uint32_t next[2][2];      // nextstate = FSM[state].next[input x][input y]
  uint32_t out[4][2];  // output = FSM[state].next[pwm][led]
} State_t;

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

```
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);
}
```

```
// Define repeat function
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;

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

  // Output State
  stateOutput();

  // blink the LED with 1 sec period (1s ON, 1s OFF)
  digitalWrite(ledPin, ledOn);

    if(ledOn == HIGH){
      if(millis() - time >= 2000){
        if(ledOn==HIGH){
        ledOn = LOW;
        }
        else{
        ledOn = HIGH;
        }
        digitalWrite(ledPin, ledOn);
        time = millis();
      }
    }
    else
    {
      ledOn = LOW;
    }

  // Print State
  Serial.print("state : ");
  Serial.print(output[1]);
  Serial.print("\t");

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

  delay(1000);
}
```

```
// Define button press function
void pressed(){
  bPressed = 1;
  nextState();
  bPressed = 0;
}
```

```
// Define nextState function
void nextState(){
  if(bPressed == 1)
  input[1] = 1;
  else
  input[1] = 0;

  state = FSM[state].next[input[1]][input[1]];

  // Output
  analogWrite(pwmPin, pwmOut);
  digitalWrite(ledPin, ledOn);
}
```

```
// Motor is operated only at short distance (<5cm)
void stateOutput(){
  if(distance < thresh){
    if(bPressed == HIGH)
    output[1] = 3;
    else
    output[1] = 1;
  }

  else
  {
    if(bPressed == HIGH)
    output[1] = 2;
    else
    output[1] = 0;
  }

    // get nextState
    pwmOut = FSM[state].out[output[1]][output[0]];
    ledOn = FSM[state].out[output[1]][output[2]];
}
```

IV. Result and Analysis

Result

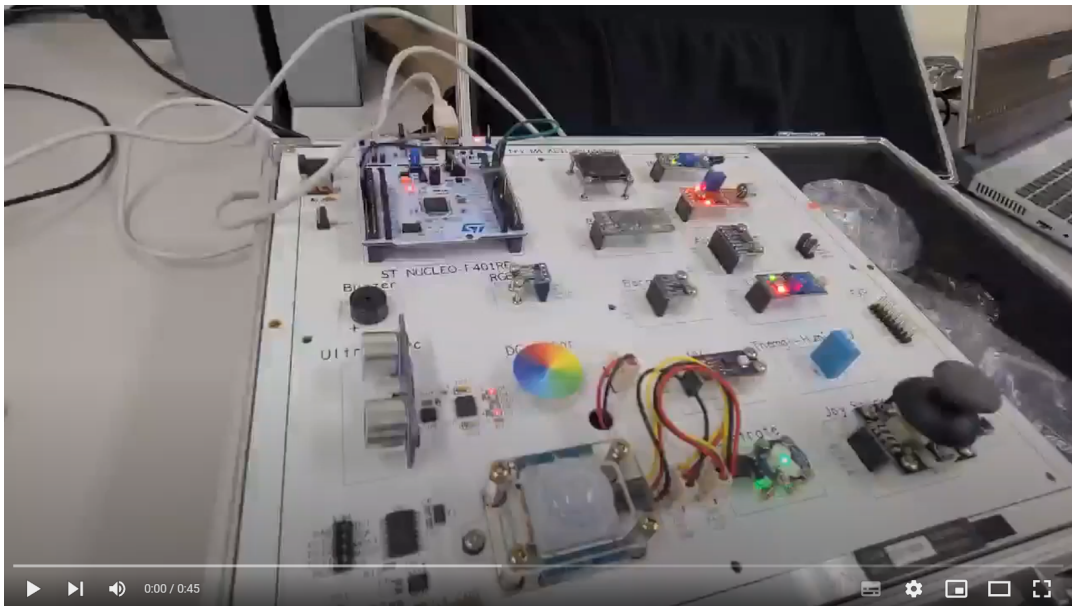
If an object is detected within 5 cm of the ultrasonic, the DC motor is operated. However, if the button is never pressed, the DC motor does not work, and there is no blinking. Once the button is pressed, the speed of the DC motor runs up to 50%. However, the motor does not work if it is more than 5cm away from Ultrasonic.

Pressing the button one more time increases the speed of the DC motor to 100%. Similarly, if an object is more than 5cm away from the ultrasonic, the motor will not operate.

If you press the button one more time, the motor speed becomes 0, and the blinking disappears. Additionally, blinking turns on for a second and turns off for a second.

Demo Video

Link : https://youtu.be/FbGgnJGKj8k



Analysis

A state table was prepared according to the conditions given in the problem. After that, since the code was implemented, if a mistake is made in writing the state table, the code is not normally implemented. However, in this problem, since the values of the state table were assigned to the array, the problem was solved by changing the values of the array.

The double arrangement was used to control the speed and blinking.

In the case of blinking, the time of blinking could be adjusted using millis and delay in the code. The delay seemed to control the blinking normally, but I could see the delay as a whole.

The state was adjusted through buttons and ultrasounds so that the desired result value was obtained. The speed was adjusted by assigning a state value to the array.

Reference

- Young-Keun Kim (2023). https://ykkim.gitbook.io/ec/