LAB: Smart mini-fan with STM32-duino

Data: 2022.09.22

Author/Partner: Seongjun Park / Junyong Choi

Demo Video: https://youtu.be/S6Lo8mnT3PA

Introduction

In this lab, I make a simple program for Smart mini-fan. The Smart mini-fan adjusts the strength of the motor every time the button is pressed, and stops operating when the object is out of a certain distance. Even if the operation is stopped, the fan is turned on through the LED.

Requirement

Hardware

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

Software

- Arduino IDE

Problem

Procedure

1. As the button **B1** is pressed, change the fan velocity.
   - MODE(state): **OFF(0%), MID(50%), HIGH(100%)**
2. When the object(face) is detected about 50mm 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
3. When the object(face) is detected within 50mm, then it automatically runs the fan
   - It must run at the speed of the current MODE
4. LED(**LED1**): Turned OFF when MODE=OFF. Otherwise, blink the LED with 1 sec period (1s ON, 1s OFF)
5. Print the distance and PWM duty ratio in Tera-Term console(every 1 sec).
6. Mearly FSM to control the mini-fan
   - Draw a FSM(finite-state-machine) table and state diagram

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.

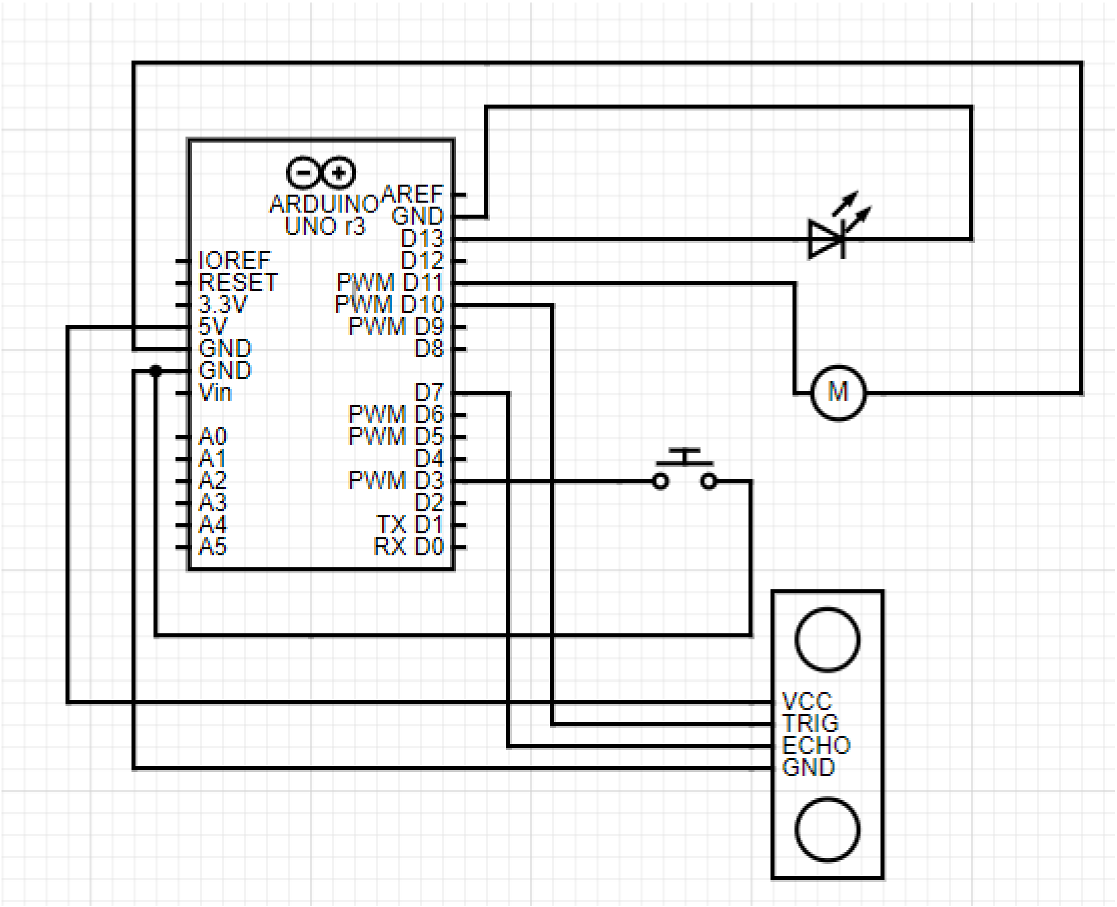
USART

- 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)

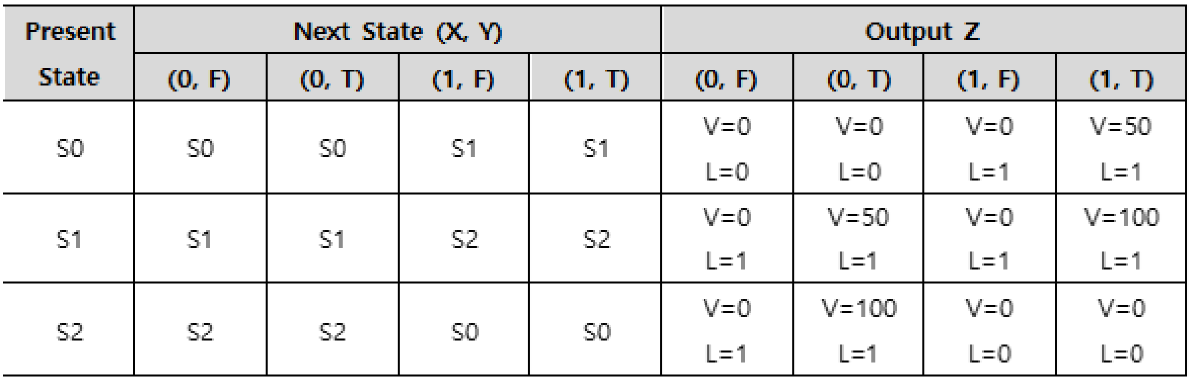
Circuit/Wiring Diagram



Algorithm

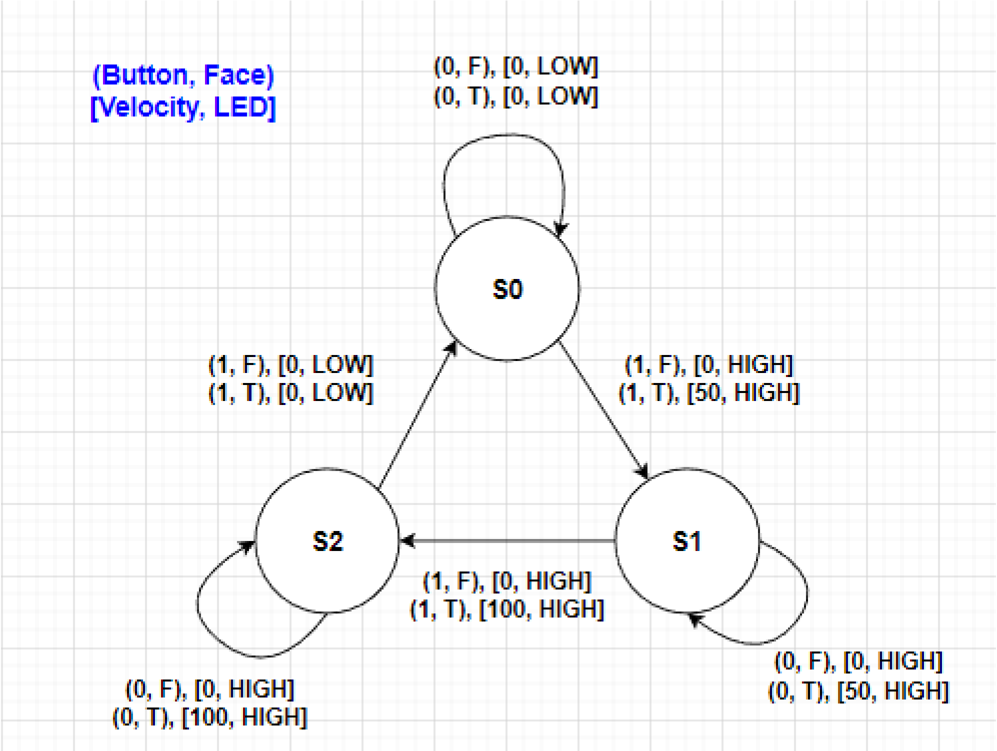
Overview

Mearly FSM 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

Mearly State Diagram



Description with Code

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

// Address number of output in array
#define PWM 0
#define LED 1

const int ledPin = 13;
const int pwmPin = 11;
const int btnPin = 3;
const int trigPin = 10;
const int echoPin = 7;

unsigned char state = S0;
unsigned char nextstate = S0;
unsigned char dInput = 1;
unsigned char bInput = 0;
unsigned char ledOut = LOW;
unsigned char pwmOut = 0;

unsigned long duration;
float distance;
int thresh = 5;

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

State_t FSM[3] = {
  { {{{0  , LOW }, {0, HIGH}},{{0 , LOW }, {125, HIGH}}}, {S0, S1} },
  { {{{0, HIGH}, {0  , HIGH }},{{125, HIGH}, {250  , HIGH }}}, {S1, S2} },
  { {{{0, HIGH}, {0  , LOW }},{{250, HIGH}, {0  , HIGH }}}, {S2, 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);
  duration = pulseIn(echoPin, HIGH);
  distance = (float)duration / 58.0;
  nextState();

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

  Serial.print("distance = ");
  Serial.print(distance);
  Serial.println(" [cm]");
  Serial.print("Duty ratio = ");
  Serial.print(pwmOut*100/255);
  Serial.println(" [%]");
  delay(1000);

  if(ledOut == HIGH)
    digitalWrite(ledPin, !digitalRead(ledPin));
  else
    digitalWrite(ledPin, LOW);

  Serial.print("distance = ");
  Serial.print(distance);
  Serial.println(" [cm]");
  Serial.print("Duty ratio = ");
  Serial.print(pwmOut*100/255);
  Serial.println(" [%]");
  delay(1000);
  // The above code is for operating every 1 second
 }

void ledBlink(){
  digitalWrite(ledPin, !digitalRead(ledPin));
}

void pressed(){
  bInput = 1;
}

void nextState(){
  if (distance < thresh)
    dInput = 1;
  else
    dInput = 0;

  pwmOut = FSM[state].out[dInput][bInput][PWM];
  ledOut = FSM[state].out[dInput][bInput][LED];

  nextstate = FSM[state].next[bInput];
  state = nextstate;
  bInput = 0;
}
```

Results and Analysis

Results

Smart-mini fan was implemented through Button and ultrasonic distance sensor. When the button was pressed once, the LED was turned on and blinking every second regardless of the sensor's result. The speed of the motor operated at 50% when the sensor detected an object less than 50mm, and the motor did not turn when it detected more than 50mm. When the button was pressed once more, the LED still blinks every second regardless of the sensor. The speed increased to 100% when the sensor detected an object, and the motor stopped when the sensor did not detect anything. When the button was pressed once more, both the LED and MOTOR were turned off regardless of the sensor.

Demo Video

https://youtu.be/S6Lo8mnT3PA

Analysis

The key was to draw the state definition, state diagram, and state table well. No matter how complicated the algorithm is, it could be fully implemented by just defining the state and drawing the table and diagram.

Reference

Young-Keun Kim. *Gitbook*. https://ykkim.gitbook.io/ec/course/lab/lab-smart-mini-fan-with-stm32-duino [2022, September 15].