LAB: Smart mini-fan with STM32-Aduino

Date: 2023.09.14

Author / ID: EunChan Kim 21801017

Demo Video: Youtube link

Introduction

In this lab, we 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.

Requirement

H/W

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

S/W

- Arduino IDE

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) areMODE(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 fanIt 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 diagramExample 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

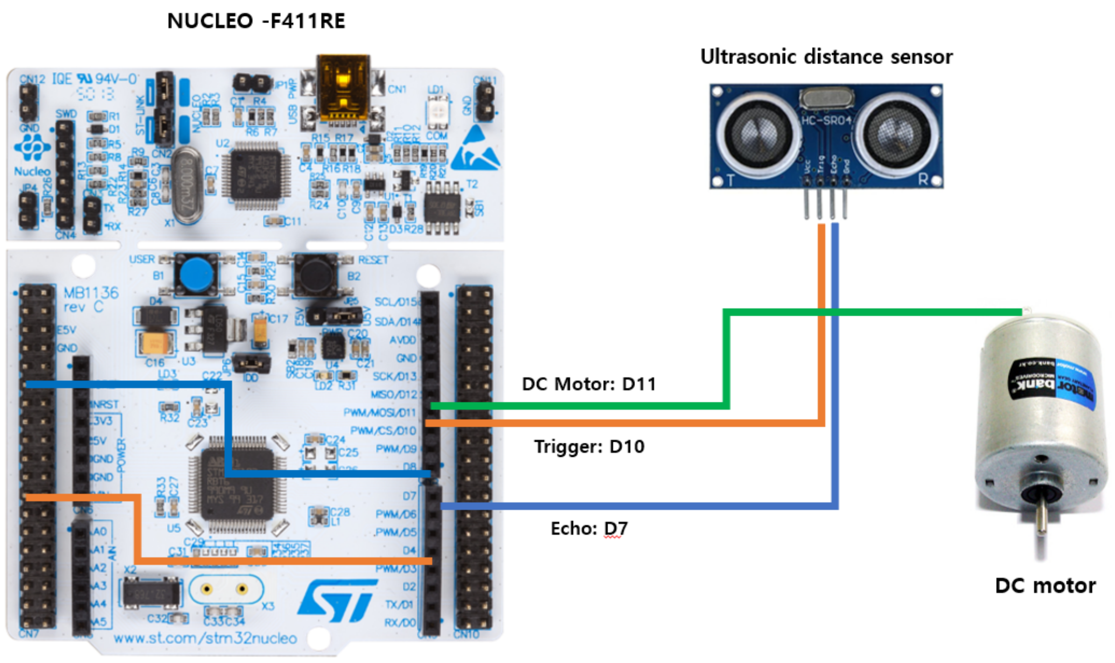


Figure1. Circuit/Wiring Diagram

Algorithm

Mealy 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

Table1. Mealy FSM TAble

Description with Code

Initialization and initial pin setting were carried out, and a State table identical to Table 1 was defined.

```
// State definition
#define S0  0   // Fan OFF
#define S1  1   // Fan vel = 50%
#define S2  2   // Fan vel = 100%

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

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

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}} }
};

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

// initialization
```

```
uint32_t state = S0;
uint32_t input[2] = {0, 0};
uint32_t pwmOut = 0;
uint32_t ledOut = LOW;

unsigned long duration;
float distance;
int thresh = 5;
unsigned long time = 0;
```

The pin modes for the LED, PWM, and ultrasonic sensor have been configured, the button's functionality has been defined, and serial communication has been set up.

```
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);
}
```

The loop iterates while the ultrasonic sensor operates, configuring the next state and output state. The results are then displayed as messages in Tera Term.

```
void loop() {
  // Generate pwm signal 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 of states
  stateOutput();

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

  // Print the message
  Serial.print("Input : ");
  Serial.print(input[1]);
  Serial.print(", ");
  Serial.print("State : ");
  Serial.print(state);
  Serial.print(", ");
  Serial.print("distance = ");
  Serial.print(distance);
  Serial.println(" [cm]");

  delay(1000);
}
```

When the distance information obtained through the ultrasonic sensor becomes 5cm or less, the next state input is set, and then the state transition is executed.

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

  // get next state
  state = FSM[state].next[input[0]][input[1]];
}
```

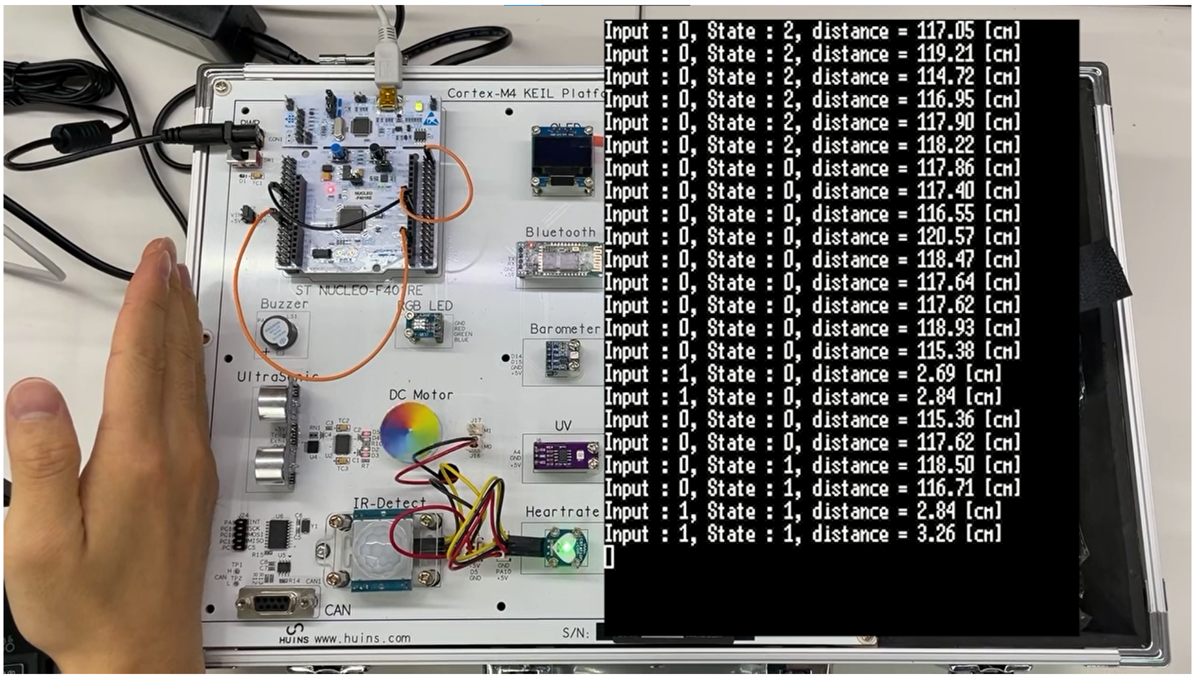
When the state is S0, meaning it's in the OFF state, I configured it to turn off the LED. In all other situations, when it's functioning normally, I made the LED blink at 1-second intervals. This function controls the LED to blink based on the current state and sets the PWM output accordingly. The PWM output is determined based on the current state and input in the state machine.

```
void stateOutput(){
  // Blink LED
  if (state == S0){
    ledOut = 0;
  }
  else{

    if (millis() - time >= 1000){
      if(ledOut == HIGH)
        ledOut = LOW;
      else
        ledOut = HIGH;

      time = millis();
    }
  }


  pwmOut = FSM[state].out[input[0]][input[1]];
}
```

Results and Analysis

Results

Utilizing Mealy FSM in this manner, all the intended functionalities were successfully achieved from the outset. Upon each button press, the motor is operated at speeds of 0%, 50%, and 100%, with the motor running exclusively when the object is in close proximity to the ultrasonic sensor. In the absence of button presses, the motor enters a pause state when the object moves away, and upon resuming operation, it maintains the previous motor speed.



```
Input : 0, State : 2, distance = 117.05 [cm]
Input : 0, State : 2, distance = 119.21 [cm]
Input : 0, State : 2, distance = 114.72 [cm]
Input : 0, State : 2, distance = 116.95 [cm]
Input : 0, State : 2, distance = 117.90 [cm]
Input : 0, State : 2, distance = 118.22 [cm]
Input : 0, State : 0, distance = 117.86 [cm]
Input : 0, State : 0, distance = 117.40 [cm]
Input : 0, State : 0, distance = 116.55 [cm]
Input : 0, State : 0, distance = 120.57 [cm]
Input : 0, State : 0, distance = 118.47 [cm]
Input : 0, State : 0, distance = 117.64 [cm]
Input : 0, State : 0, distance = 117.62 [cm]
Input : 0, State : 0, distance = 118.93 [cm]
Input : 0, State : 0, distance = 115.38 [cm]
Input : 1, State : 0, distance = 2.69 [cm]
Input : 1, State : 0, distance = 2.84 [cm]
Input : 0, State : 0, distance = 115.36 [cm]
Input : 0, State : 0, distance = 117.62 [cm]
Input : 0, State : 1, distance = 118.50 [cm]
Input : 0, State : 1, distance = 116.71 [cm]
Input : 1, State : 1, distance = 2.84 [cm]
Input : 1, State : 1, distance = 3.26 [cm]
```

Figure2. Mini-Fan Lab Result

Analysis

As evident from the demo video, all experiment requirements have been met. However, there is suspicion of a double-button press occurrence during button press actions. Intermittently, there has been an occurrence of the state transitioning by two steps at a time. It appears necessary to investigate whether this issue is hardware-related or a result of the code structure.

Reference

- **LAB: Smart mini-fan with STM32-aduino by YK Kim (gitbook Link)**

Appendix

Entire Code

```
/*---------------------------------------------------------------------------------
\
@ [LAB] Arduino-STM32 mini-fan
Author           : EunChan Kim
Created          : 08-09-2023
Modified         : 14-09-2023
Language/ver     : Arduino IDE 2.2.0
-------------------------------------------------------------------------------
*/

// State definition
#define S0  0   // Fan OFF
#define S1  1   // Fan vel = 50%
#define S2  2   // Fan vel = 100%

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

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

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}} }
};

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

// initialization
uint32_t state = S0;
uint32_t input[2] = {0, 0};
uint32_t pwmOut = 0;
uint32_t ledOut = LOW;

unsigned long duration;
float distance;
int thresh = 5;
unsigned long time = 0;

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 signal 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 of states (Mealy style)
  stateOutput();

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

  // Print the message
  Serial.print("Input : ");
  Serial.print(input[1]);
  Serial.print(", ");
  Serial.print("State : ");
  Serial.print(state);
  Serial.print(", ");
  Serial.print("distance = ");
  Serial.print(distance);
  Serial.println(" [cm]");

  delay(1000);
}

// When button pressed
void pressed(){
  input[0] = 1;
  stateOutput();
  nextState();
  input[0] = 0;
}
```

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

  // get next state (Mealy style)
  state = FSM[state].next[input[0]][input[1]];
}

void stateOutput(){
  // Blink LED
  if (state == S0){
    ledOut = 0;
  }
  else{

    if (millis() - time >= 1000){
      if(ledOut == HIGH)
        ledOut = LOW;
      else
        ledOut = HIGH;

      time = millis();
    }
  }


  pwmOut = FSM[state].out[input[0]][input[1]];
}
```