Arduino Overview. Ultrasound, Termometer and Fan control

Javier Asensio, Marc Dalmasso, José Pedro Conceição

October 2018

Abstract

The goal of this practice is to work with and to introduce ourselves to Arduino's programming and devices lecture and control. Code for controlling three different devices, both independently and interacting, have been developed. These three devices are: ultrasound distance sensor, temperature sensor and a CPU fan containing an encoder, making it possible to read the actual fan speed, enabling closed loop operations.

1 Ultrasound

In this first step, only the ultrasound distance sensor is used. The given device *HC-SR04* requires being given connected to power supply (vcc and GROUND pins), is controlled through the trigger signal (TRG pin) and gives its output as a delayed response on the ECHO pin. As a brief behaviour explanation: as the device recieves the trigger pulse through the TRG pin, the device generated a 8 cycle of ultrasonic burst at 40kHz and sets the ECHO pin to 1. Once the burst is received again when it is reflected by an object (in case there is an object or this object is enough close to the sensor) the ECHO pin is set to 0. If no signal is recieved, the echo pin is set to 0 in a given time. The duration of the activation of this pin is directly proportional to the distance between the object and the sensor.

The features of this sensor are the following:

- Power supply: +5V DC.
- Range: 2cm – 400cm.
- Resolution: 0.3 cm.
- Output: Digital, pulse of variable width.

The code developed to make work correctly the system is the following:

```
int vccPin = 2; int gndPin = 5;
int trgPin = 3; int echoPin = 4;
int inPin[] = {echoPin};
int outPin[] = {vccPin,trgPin,gndPin};
unsigned long pulsetime;
unsigned long distance;


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  Serial.begin(9600);
  for(int i = 0; i < sizeof(inPin)/sizeof(inPin[0]); i++) {
    pinMode(inPin[i],INPUT);
  }
  for(int i = 0; i < sizeof(outPin)/sizeof(outPin[0]); i++) {
    pinMode(outPin[i],OUTPUT);
  }
  digitalWrite(vccPin, HIGH);
  digitalWrite(gndPin, LOW);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(trgPin,LOW);
  delayMicroseconds(5);
  digitalWrite(trgPin,HIGH);
  delayMicroseconds(10);
  digitalWrite(trgPin,LOW);
  pulsetime = pulseIn(echoPin,HIGH);
  distance = pulsetime/58;
  if(distance < 3300){
     Serial.print(distance);
  Serial.print("cm\n");
  }

}
```

It can be seen that once it is obtained the duration of the ECHO signal mode ON, it is divided by 58 in order to obtain the distance. It is applied also a threshold of 3300 because when the distance is over this value, the sensor is not able to get a good approximation of the distance.

2 Thermometer

In this part, the code for the temperature sensor *LM35* reading is developed. This sensor has 3 pins: two for the power supply (+Vs and GND) and the third one, which generates the output voltage. This output voltage is proportional to the temperature, defined by the given equation: $V_{out}(mV) = 10 * T(^oC)$.

The features of this temperature sensor are the following:

- Power supply: 4V-20V.
- Range: -55 to +155°C.

- Accuracy: 0.5ºC (at 25ºC).
- Output: Analog, linear.

2.1 Corrected Ultrasound

In this code, as the temperature is obtained the correction for the ultrasound sensor has been made, given that the speed of the bursts will change depending on the temperature. For this case, the corrective formula would be: $speedOfSound = 331.3+0.606*temp$; and the distance compensated will be: $DistanceCompensated = (pulsetime/2000)*speedOfSound$.

```


//Temperature Sensor
int vsPinA = A0; int voutPinA = A1; int gndPinA = A2;

//UltraSound Sensor
int vccPin = 2; int gndPin = 5;
int trgPin = 3; int echoPin = 4;

int inPin[] = {echoPin, voutPinA};
int outPin[] = {vccPin, trgPin, gndPin, vsPinA, gndPinA};
unsigned long pulsetime;
unsigned long voltage;


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  Serial.begin(9600);
  for (int i = 0; i < sizeof(inPin) / sizeof(inPin[0]); i++) {
    pinMode(inPin[i], INPUT);
  }
  for (int i = 0; i < sizeof(outPin) / sizeof(outPin[0]); i++) {
    pinMode(outPin[i], OUTPUT);
  }
  digitalWrite(vccPin, HIGH);
  digitalWrite(gndPin, LOW);
  analogWrite(vsPinA, 255 );
  analogWrite(gndPinA, 0);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(trgPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trgPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trgPin, LOW);
  pulsetime = pulseIn(echoPin, HIGH);
  float distance = pulsetime / 58;
  if (distance < 3300) {
    Serial.print(distance);
    Serial.print("cm  ");
    voltage = analogRead(voutPinA);
    float temp = voltage/2.55;
    Serial.print(temp);
    Serial.print("\ang{C}");

    float speedOfSound = 331.3 + 0.606 * temp;
    Serial.print("sound v: ");
    Serial.print(speedOfSound);
    Serial.print(" m/s   ");

    float compensatedDistance = (pulsetime / 20000.0) * speedOfSound;
    //duration is in us, speed in m/s. /20k gives cm
    Serial.print("comp. dist: ");
    Serial.print(compensatedDistance, 1);         //one decimal
    Serial.print(" cm\n");
  }

}
```

3 Fan

In this last step, the third device, the *QFR0912DE-F00* fan, is added. This fan works at a nominal frequency of 25kHz. The main characteristics are the following:

- Dimensions: 80*80*38mm
- Voltage supply: +12V DC
- Input current: 1.15A nominal, 1.70A max.
- Speed: 9000 RPM max
- Max. air flow: 105 CFM (cubic feet per minute)
- PWM input: 25kHz nominal frequency
- Integrated speed sensor

And the input and output ports are the following:

- Red: +12V DC.
- Black: GND.
- Yellow: PWM input.
- Blue: F00 - speed sensor output.

As it can be seen in the code, it is asked in the terminal for a specific value of the fan speed and the desired given speed is applied by controlling the amount of time the fan is turned on. In this way, and always in a loop, the fan will have a range of frequencies, from 0kHz to 25kHz.
Taking into account that the speed written by the user will be the variable fspeed,

the formula applied to compute the alive time is $fspeed * 40$ microseconds, setting the fan as HIGH (1); and $1000 - fspeed * 40$ microseconds for the time when the fan is LOW (0). This past computations are applied on the microcontroller loop, having it a frequancy of 1 kHz. For example when the user wants the 25kHz (max velocity), the time per millisecond the fan is HIGH will be 1000 (microseconds) and then the time when is LOW will be 0. So the fan will be always activated in order to get its maximum speed. If fspeed is less than 25, the fan will have a ranges of HIGH and LOW which allows to modify the velocity of this.

```
//Fan
int PWMPin = 7;
int F00Pin = 11;
int fspeed = 0;

int inPin[] = {F00Pin};
int outPin[] = {PWMPin};

void setup() {
  Serial.begin(9600);
  for (int i = 0; i < sizeof(inPin) / sizeof(inPin[0]); i++) {
    pinMode(inPin[i], INPUT);
  }
  for (int i = 0; i < sizeof(outPin) / sizeof(outPin[0]); i++) {
    pinMode(outPin[i], OUTPUT);
  }

}

void loop() {
  digitalWrite(PWMPin, HIGH);
  delayMicroseconds(fspeed*40);
  //delayMicroseconds(600);
  digitalWrite(PWMPin, LOW);
  delayMicroseconds(1000-fspeed*40);
  //delayMicroseconds(400);
  if (Serial.available() > 0) {
    int aux = Serial.parseInt();
    if (aux > 0){
      fspeed = min(aux,25);
      Serial.print(fspeed);
    }

  }

}
```