

/*

//The drone altitude control program uses a potentiometer and readings from a HC-SR04 ultrasonic sensor to control the altitude of a DIY mini drone. Readings from the ultrasonic sensor are used to determine distance from an object. This value is then used in an equation with a potentiometer to proportionally control how much power is given to the drone motors.

Project instructions for DIY mini drone available at:

<https://www.sciencebuddies.org/stem-activities/diy-mini-drone-arduino-altitude-control>

Ben Finio, Science Buddies

2021

The below code was adopted from Science Buddies and adjusted to work with a HC-SR04 ultrasonic sensor instead of a Ping, which is used in the project being referenced. Work by Jean Williams and Vanessa Begat, 2022

*/

// Global variables are defined below according

// constants that don't change (the specific pin locations are set to how I wired it for demo on 6.27.22)

const int trigPin = 11; // pin for the trig setting on the HC-SR04 (this is labeled on the sensor)

const int echoPin = 9; // pin for the echo setting on the HC-SR04 (labeled on the sensor; trig and echo must be plug into PWM ~ pins)

const int PWMPin = 10; // pin for the motor PWM signal (this is how we will send the signal to control the motor speed)

const int but_pin = 3; // pin for start/stop button

const int potPin = A0; // pin for potentiometer

const int min_height = 10; // minimum desired height in centimeters for the drone

const int max_height = 50; // maximum desired height in centimeters for the drone

const float K = 2; // proportional control gain (the correction factor)

// lower K value if drone is bouncing bc of code; increase if it's too slow to adjust to a height

const int ramp_time = 40; // delay provided for motors to ramp up/ramp down

// variables that can change

int but_status = 1; // button status

int target_dist = 0; // target distance in centimeters (this will be set with potentiometer)

int pot_reading = 0; // potentiometer reading

long duration; // duration of the ultrasonic ping (a measure of how long the echo pin is engaged in HIGH)

long cm; // measured distance in centimeters (the duration time will be converted into cm)

int PWM_offset = 180; // offset for PWM signal (between 0-255), roughly the value at which the drone hovers
// You will need to tweak this value if having trouble with drone not lifting off

int PWM_signal = 0; // PWM value

int error = 0; // difference between target distance and measured distance

void setup() { // setup code that only runs once

pinMode(but_pin, INPUT); // Set button pin to input. You can omit the external pullup resistor and use INPUT_PULLUP instead.

pinMode(trigPin, OUTPUT); // The trigPin is what is used to turn on the echo signal on the HC-SR04, which is why it is an output.

```

pinMode(echoPin, INPUT); // The echoPin is where the ultrasonic sensor data originates on the HC-SR04.
Serial.begin(9600);      // initialize serial communication, use for debugging if needed. Select Tools --> Serial
Monitor to open window.
Serial.print("Press button to start"); // You will see this command on the serial monitor at the beginning of
program initiation.
WaitForPress(); // wait for button to be pressed to start ramping up the motors for lift off
LiftOff();      // give drone a short boost to get off the ground and prevent bad sensor readings
delay(250);     // wait briefly before entering loop again to get better sensor readings
}

void loop() {

    but_status = digitalRead(but_pin); // read button pin (1st button press starts the drone; 2nd press initiates
drone landing)
    if(but_status == LOW){
        Land(); // land slowly to avoid damage (motors will receive a slow reduction of power rather than abrupt
cutoff)
        delay(1000); // wait for one second, then start monitoring for another button press for another liftoff
        but_status = digitalRead(but_pin); // resets the value of the pin
        WaitForPress(); // wait for button to be pressed to start again
        // give drone a short boost to get off the ground and prevent bad sensor readings
        LiftOff();
    }

    // HC-SR04 is triggered by a HIGH pulse of 2 or more microseconds.
    // below code makes sure the sensor is off to start with, gives a delay, and then turns it on via the trigPin
turning on echoPin straight to HIGH
    // echo pin will read HIGH until it detects the returned signal that bounces off an object
    // you have to turn the trigPin back to LOW so that you can pulse it back to HIGH to make sure the echoPin
continues to take readings
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    duration = pulseIn(echoPin, HIGH); // line of code used to read the duration that echoPin registers as HIGH at
one time. The result is in microseconds.
    // The speed of sound is 340 m/s or 29 microseconds per centimeter.
    // The echo ping travels out and back and is detected by the echoPin.
    // To find the object's distance away from the sensor, we divide the distance traveled by 2.

    cm = duration / 29 / 2; //speed of sound wave divided by 2

    // below code is used to read the potentiometer and then use proportional control to guide the drone

    pot_reading = analogRead(potPin); // Read analog value from the potentiometer. This
returns a value between 0-1023

```

```

    target_dist = map(pot_reading,0,1023,min_height,max_height); // Map the potentiometer reading to a height
    between min_height (0) and max_height (1023)
    error = target_dist - cm; // line of code to calculate difference between target height and
    actual height
    PWM_signal = K*error+PWM_offset; // calculate PWM value using proportional controller
    // Predetermined correction factor x difference in height + the PWM
    value the drone hovers at)
    if(PWM_signal>255){ // make sure PWM signal does not exceed allowed range of
    0-255
        PWM_signal = 255;
    }
    else if(PWM_signal<0){
        PWM_signal = 0;
    }

    analogWrite(PWMPin, PWM_signal); // send PWM signal to output pin

    // Print information to serial monitor for debugging purposes
    // Comment out this code if you don't need it
    Serial.print("Target distance: ");
    Serial.print(target_dist);
    Serial.print(" Measured distance: ");
    Serial.print(cm);
    Serial.print(" Error: ");
    Serial.print(error);
    Serial.print(" PWM: ");
    Serial.print(PWM_signal);
    Serial.print(" Button Status: ");
    Serial.print(but_status);
    Serial.println();

}

void WaitForPress(){ // wait for button to be pressed to start
    while(but_status == HIGH){ // get stuck in a loop and wait to start until the button is pressed again
        but_status = digitalRead(but_pin); // read button pin
    }
}

void LiftOff(){ // slowly ramp up motor speed to lift off
    while(PWM_signal<PWM_offset){ //slowly ramp up motor speed for smooth takeoff
        analogWrite(PWMPin, PWM_signal); // send PWM signal to output pin
        PWM_signal++; // signals to give more power to the motors increasing lift
        delay(ramp_time);
    }
}

void Land(){ // slowly ramp down motor speed to land safely
    while(PWM_signal>0){

```

```
PWM_signal--;    // signals to give less power to the motors decreasing lift
analogWrite(PWMPin, PWM_signal);
delay(ramp_time);
}
}
```