



In Partnership With



Feedback Control Assignment

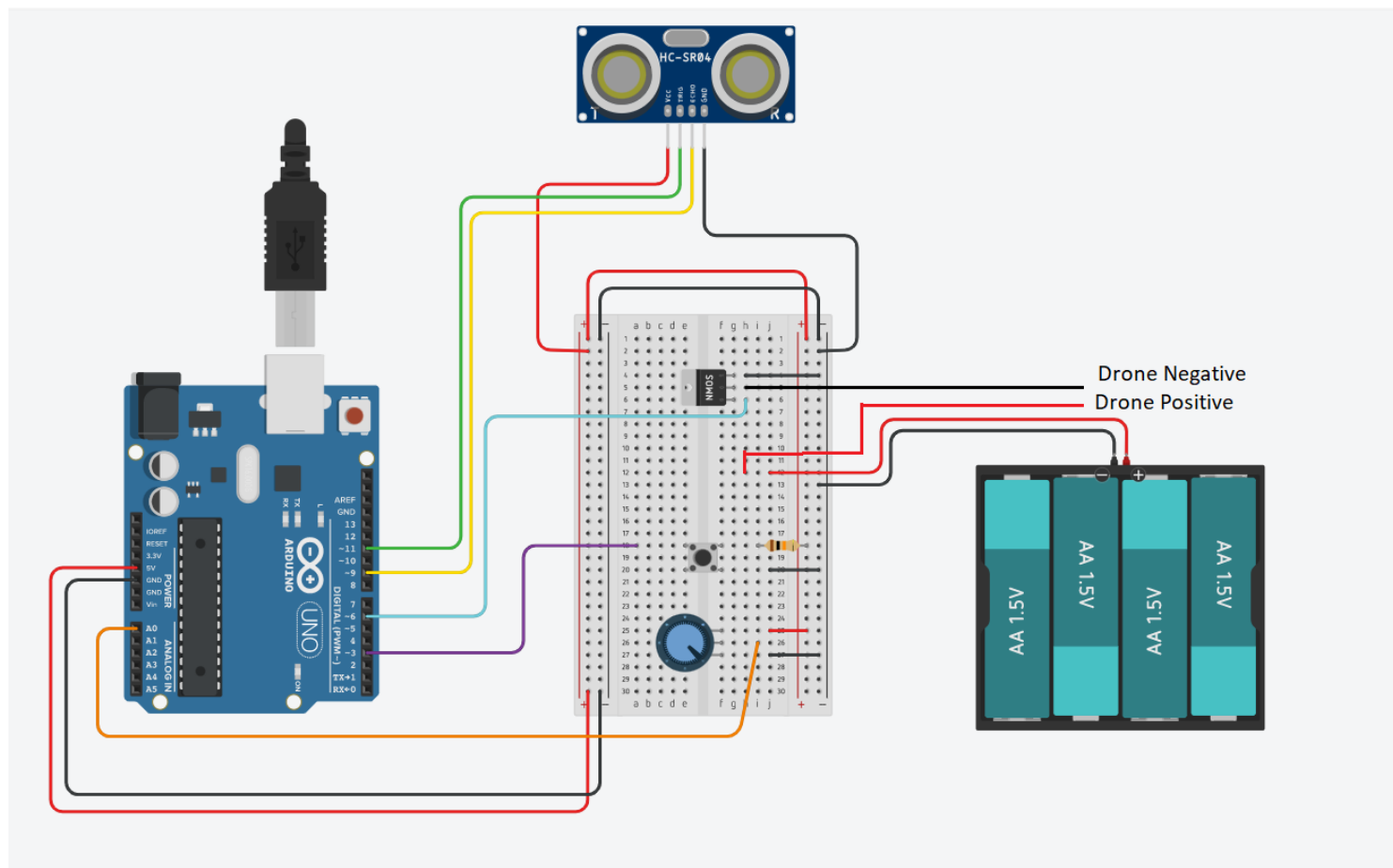
Challenge

Using the Arduino, you must create a drone whose altitude is controlled using a potentiometer. The altitude of the drone should be mapped to the values of the potentiometer so that a low value of the potentiometer means that the drone is low. A high value of the potentiometer means that the drone goes up.

After building the drone and getting it controlled via the potentiometer, you must change the programming so that the drone always stays a fixed height away from the ground without input from the potentiometer. To accomplish this, you will move your ultrasonic sensor up and down and the drone should stay a fixed distance away from the sensor.

Bonus: Have your drone fly to pre-programmed heights with no input from the human.

- Using the skeleton below, write the pseudocode for this program in your engineering notebook. Much of it is already provided to you in the form of comments, but see if you can read and interpret what is happening and create the pseudocode flowchart for the program.
- Wire your board according to the schematic below.
- Use the skeleton code that you have been provided and fill in the parts in yellow to create a drone that flies according to input from the potentiometer.



Caution: Do not assemble this circuit while your Arduino is plugged in or while your batteries are in the battery holder. DO NOT plug your battery pack into the 5V Arduino power rail.

```
// constants that don't change
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 = ; // student action: determine minimum desired height in cm for the drone based on the length of your dowel
const int max_height = ; // student action: determine maximum desired height in cm for the drone based on the length of your dowel
const float K = 2; // proportional control gain (the correction factor)
// lower K value if drone is bouncing back and forth 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 starting value
int error = 0; // difference between target distance and measured distance

void setup() { // setup code that only runs once
//student action: set up button pinMode using variables from above
//student action: set up pinMode for trigPin using variables from above
//student action: set up pinMode for echoPin using variables from above
//student action: turn on your serial monitor using variables from above

  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 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();
    }
}
```

/* The code below 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. HC-SR04 is triggered by a HIGH pulse of 2 or more microseconds.*/

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
```

/*Student action: use the pulseIn() command to read the value from your ultrasonic sensor. Store the value in a variable called "duration". Convert the value of duration into centimeters. Store the value in centimeters in a variable called "cm" */



In Partnership With

LSU
Gordon A. Cain Center
For Scientific, Technological,
Engineering & Mathematical Literacy

```
/*Student action: read the potentiometer value. Store that value in a variable called "pot_reading". Take that reading and use the  
map() function to map the potentiometer reading (0 to 1023) to the min_height (from declaration above) and max_height (from  
declaration above). Assign the newly mapped value to a variable called "target_dist"*/
```

```
//The potentiometer value will use proportional control to guide the drone.
```

```
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){  
    PWM_signal = 255;
```

```
// make sure PWM signal does not exceed allowed range of 0-255
```

```
}  
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
```

```
// Note: serial print commands are slow, you can use them initially but can comment them out once your drone is working
```

```
Serial.print("Target distance: ");
```

```
Serial.print(target_dist);
```

```
Serial.print(" Measured distance: ");
```

```
Serial.print(cm);
```

```
Serial.print(" Error: ");
```

```
Serial.print(error);
```



In Partnership With



```
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);
  }
}
```