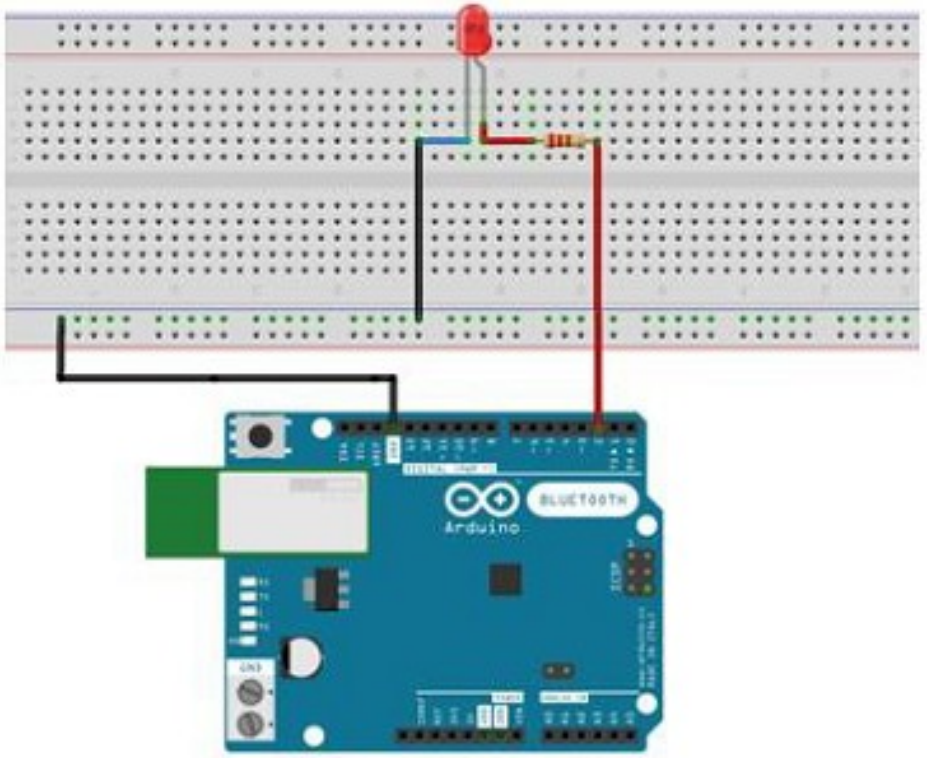
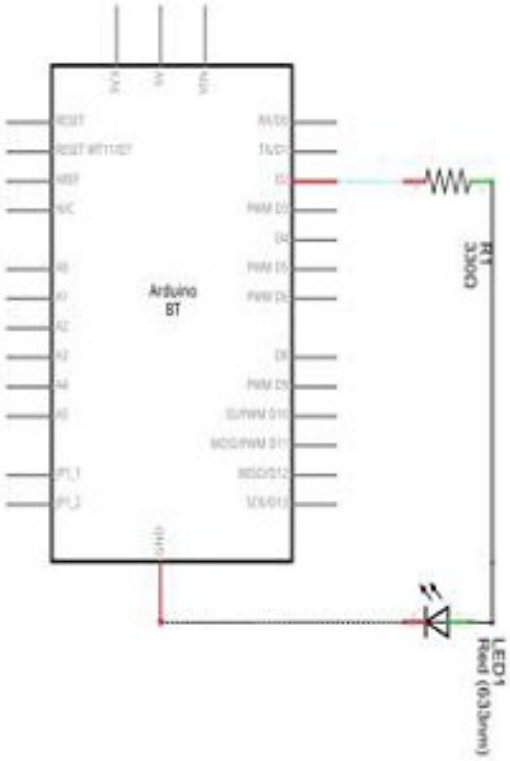
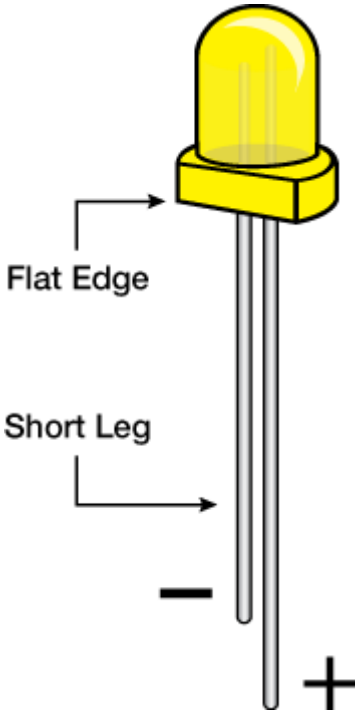
# Arduino: projects

Asst. Prof. Dr. Yalçın İŞLER

# Blinking LED: circuit



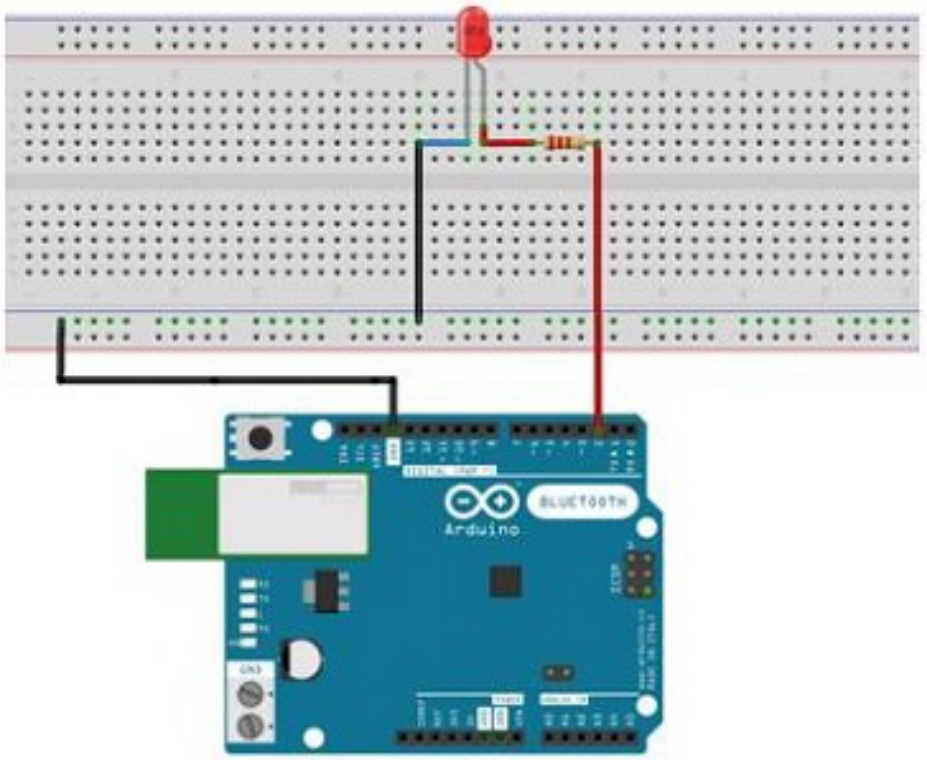



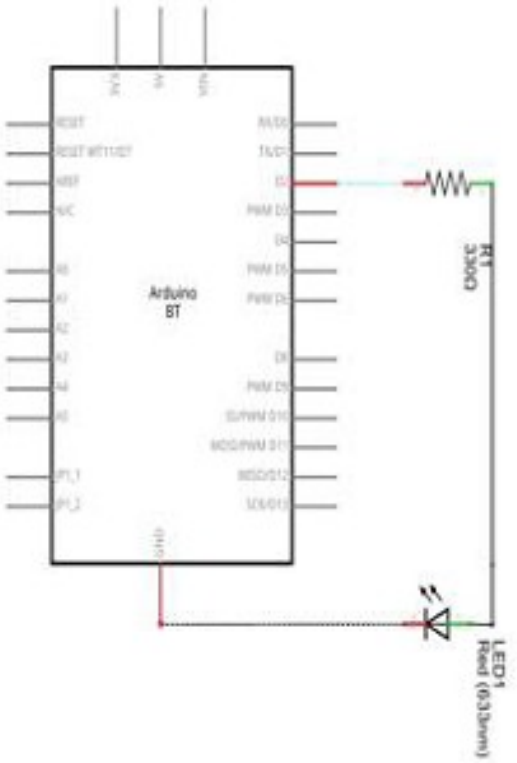



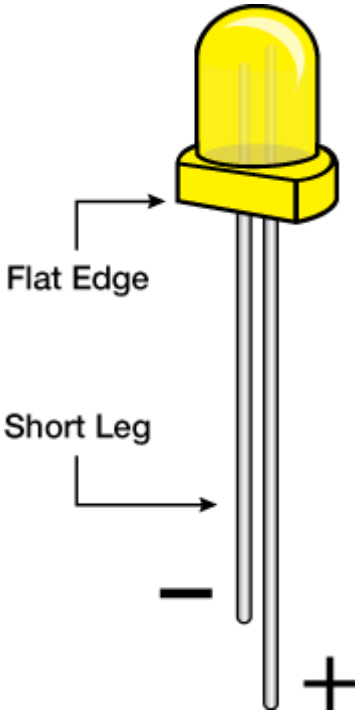
# Blinking LED: code

```
/*
   Blink
   Turns on an LED on for one second, then off for one second, repeatedly.
*/

// the setup function runs once when you press reset or power the board

void setup() {  // initialize digital pin 13 as an output.
   pinMode(2, OUTPUT);
}

// the loop function runs over and over again forever

void loop() {
   digitalWrite(2, HIGH); // turn the LED on (HIGH is the voltage level)
   delay(1000); // wait for a second
   digitalWrite(2, LOW); // turn the LED off by making the voltage LOW
   delay(1000); // wait for a second
}
```

# Fading LED: circuit, *the same :))*



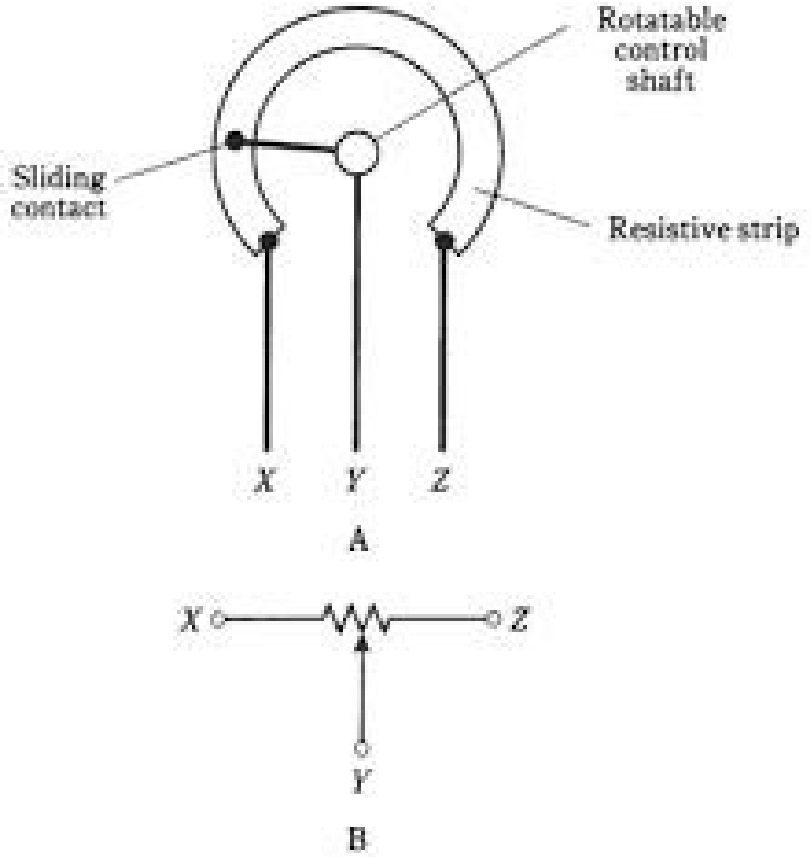



# Fading LED: code

```
/*
   Fade
   This example shows how to fade an LED on pin 9 using the analogWrite() fur

   The analogWrite() function uses PWM, so if you want to change the pin you'
   sure to use another PWM capable pin. On most Arduino, the PWM pins are ide
   a "~" sign, like ~3, ~5, ~6, ~9, ~10 and ~11.
*/

int led = 9; // the PWM pin the LED is attached to
int brightness = 0; // how bright the LED is
int fadeAmount = 5; // how many points to fade the LED by
// the setup routine runs once when you press reset:

void setup() {
   // declare pin 9 to be an output:
   pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:

void loop() {
   // set the brightness of pin 9:
   analogWrite(led, brightness);
   // change the brightness for next time through the loop:
   brightness = brightness + fadeAmount;
   // reverse the direction of the fading at the ends of the fade:
   if (brightness == 0 || brightness == 255) {
      fadeAmount = -fadeAmount ;
   }
   // wait for 30 milliseconds to see the dimming effect
   delay(300);
}
```

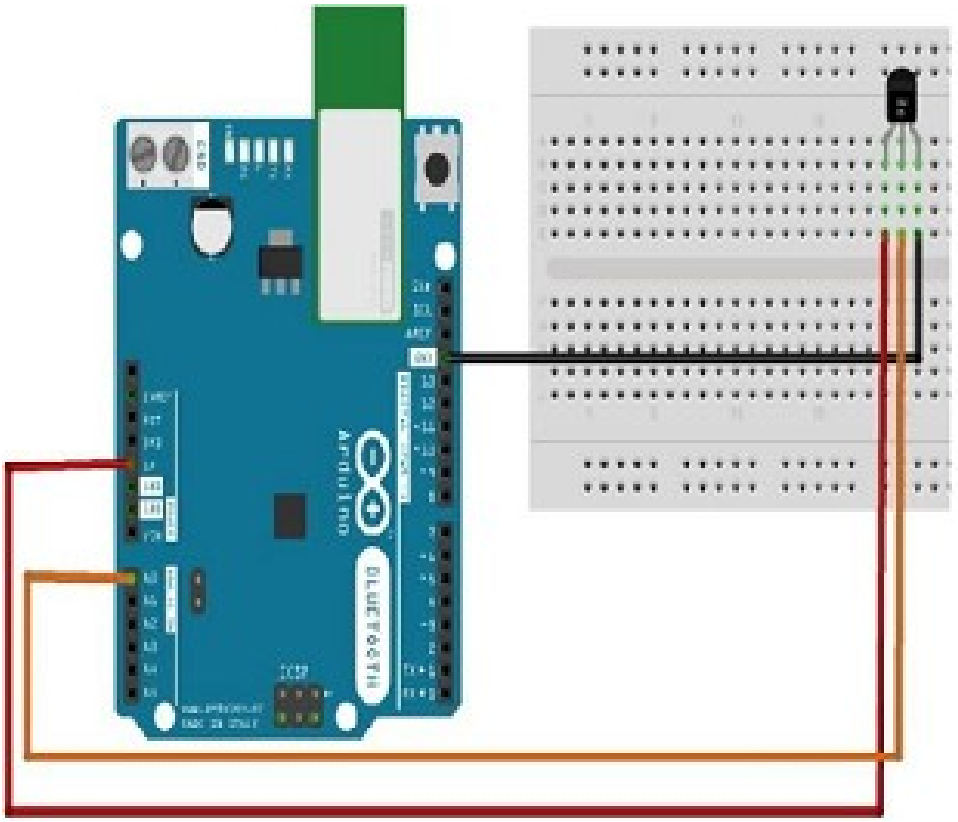
# Analog Read: circuit element

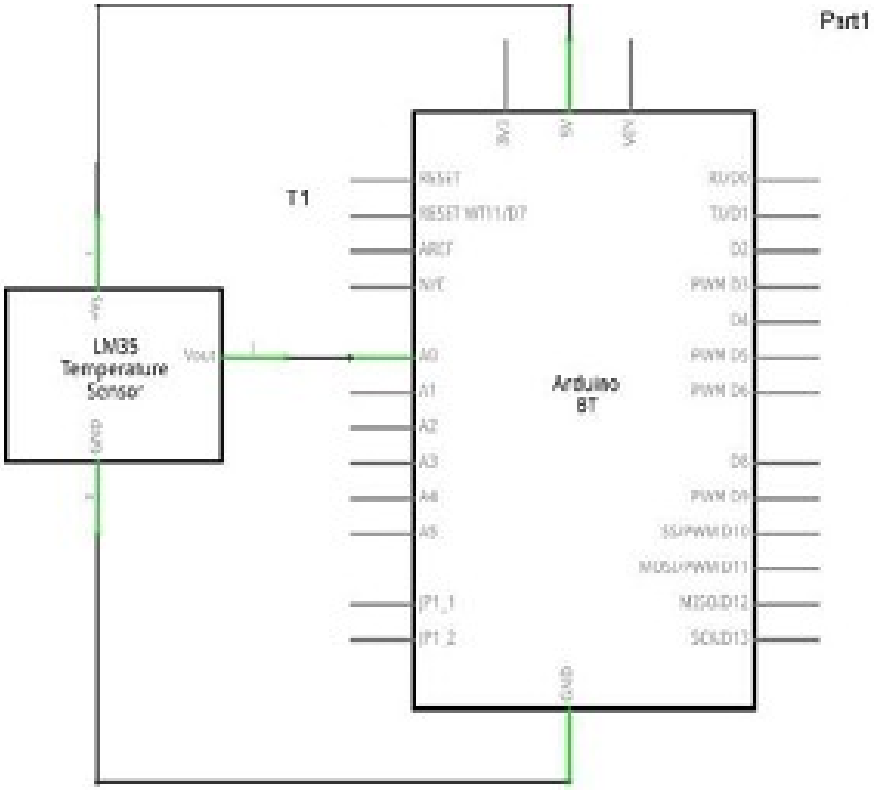



# Analog Read: code

```
/*
   ReadAnalogVoltage
   Reads an analog input on pin 0, converts it to voltage,
   and prints the result to the serial monitor.
   Graphical representation is available using serial plotter (Tools > Seria
   Attach the center pin of a potentiometer to pin A0, and the outside pins
*/

// the setup routine runs once when you press reset:

void setup() {
   // initialize serial communication at 9600 bits per second:
   Serial.begin(9600);
}

// the loop routine runs over and over again forever:

void loop() {
   // read the input on analog pin 0:
   int sensorValue = analogRead(A0);
   // Convert the analog reading (which goes from 0 - 1023) to a voltage (0
   float voltage = sensorValue * (5.0 / 1023.0);
   // print out the value you read:
   Serial.println(voltage);
}
```

# Temperature Read: circuit

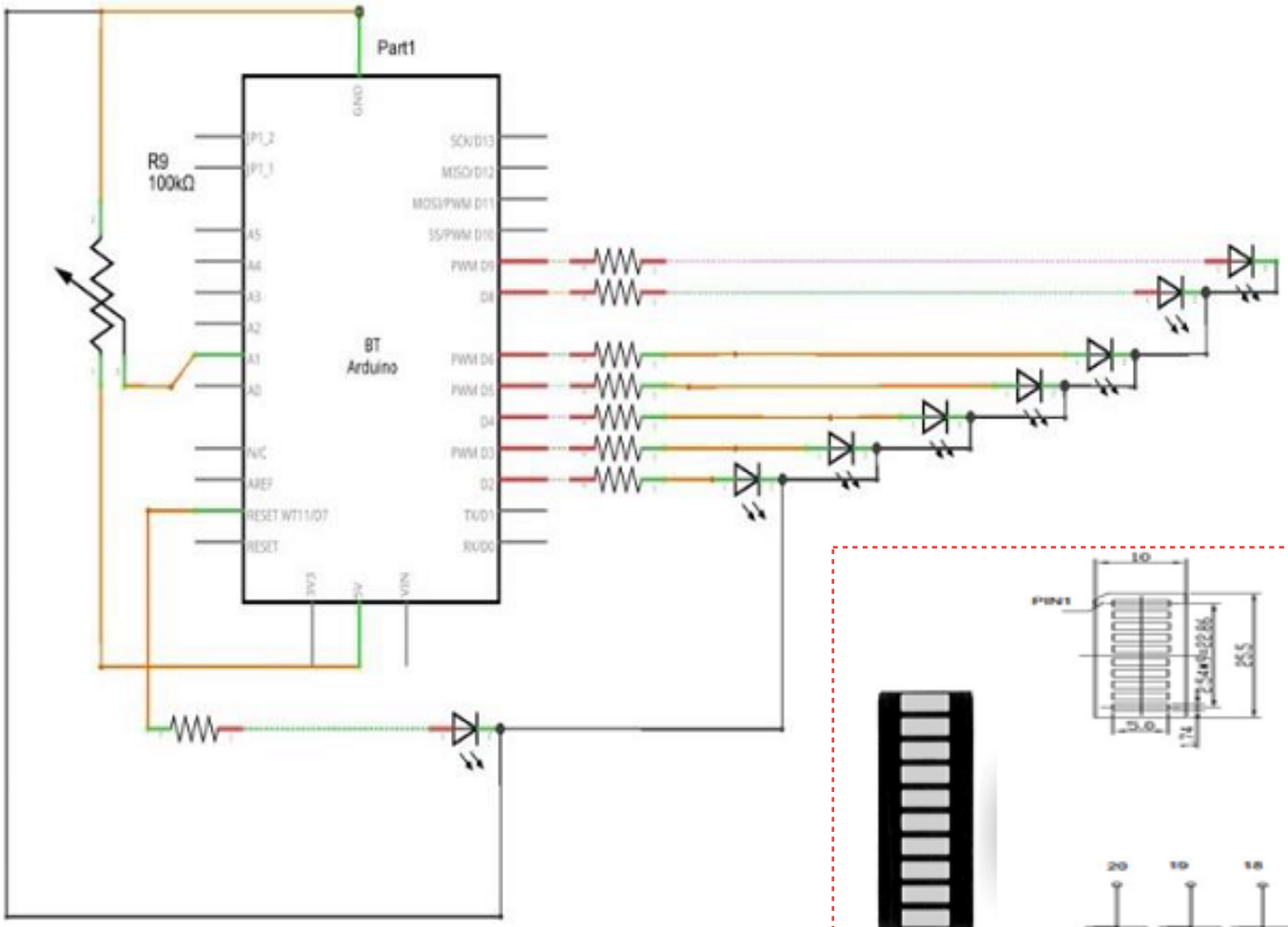


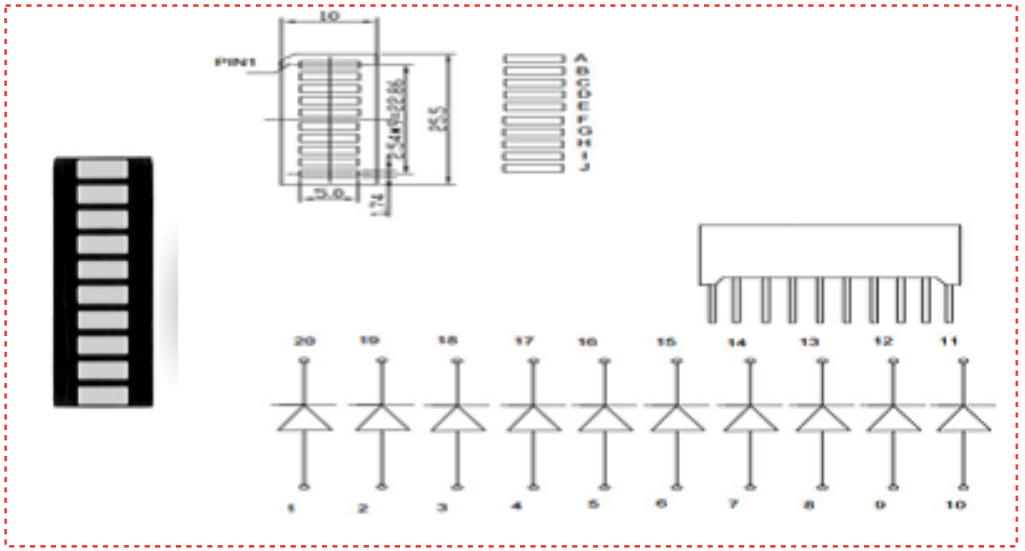


# Temperature Read: code

```
float temp;
int tempPin = 0;

void setup() {
   Serial.begin(9600);
}

void loop() {
   temp = analogRead(tempPin);
   // read analog volt from sensor and save to variable temp
   temp = temp * 0.48828125;
   // convert the analog volt to its temperature equivalent
   Serial.print("TEMPERATURE = ");
   Serial.print(temp); // display temperature value
   Serial.print("*C");
   Serial.println();
   delay(1000); // update sensor reading each one second
}
```

# LED Bar: circuit

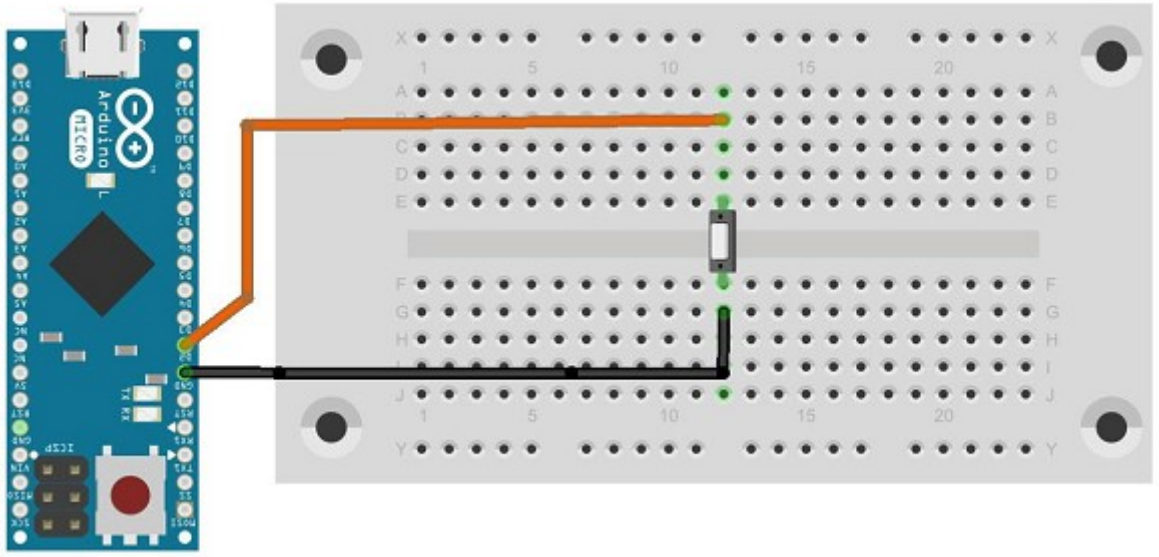


# LED Bar: code

```
// these constants won't change:
const int analogPin = A0; // the pin that the potentiometer is attached to
const int ledCount = 8; // the number of LEDs in the bar graph
int ledPins[] = {2, 3, 4, 5, 6, 7, 8, 9}; // an array of pin numbers to which

void setup() {
   // loop over the pin array and set them all to output:
   for (int thisLed = 0; thisLed < ledCount; thisLed++) {
      pinMode(ledPins[thisLed], OUTPUT);
   }
}

void loop() {
   // read the potentiometer:
   int sensorReading = analogRead(analogPin);
   // map the result to a range from 0 to the number of LEDs:
   int ledLevel = map(sensorReading, 0, 1023, 0, ledCount);
   // loop over the LED array:
   for (int thisLed = 0; thisLed < ledCount; thisLed++) {
      // if the array element's index is less than ledLevel,
      // turn the pin for this element on:
      if (thisLed < ledLevel) {
         digitalWrite(ledPins[thisLed], HIGH);
      }else { // turn off all pins higher than the ledLevel:
         digitalWrite(ledPins[thisLed], LOW);
      }
   }
}
```

# Keyboard Emulator: circuit

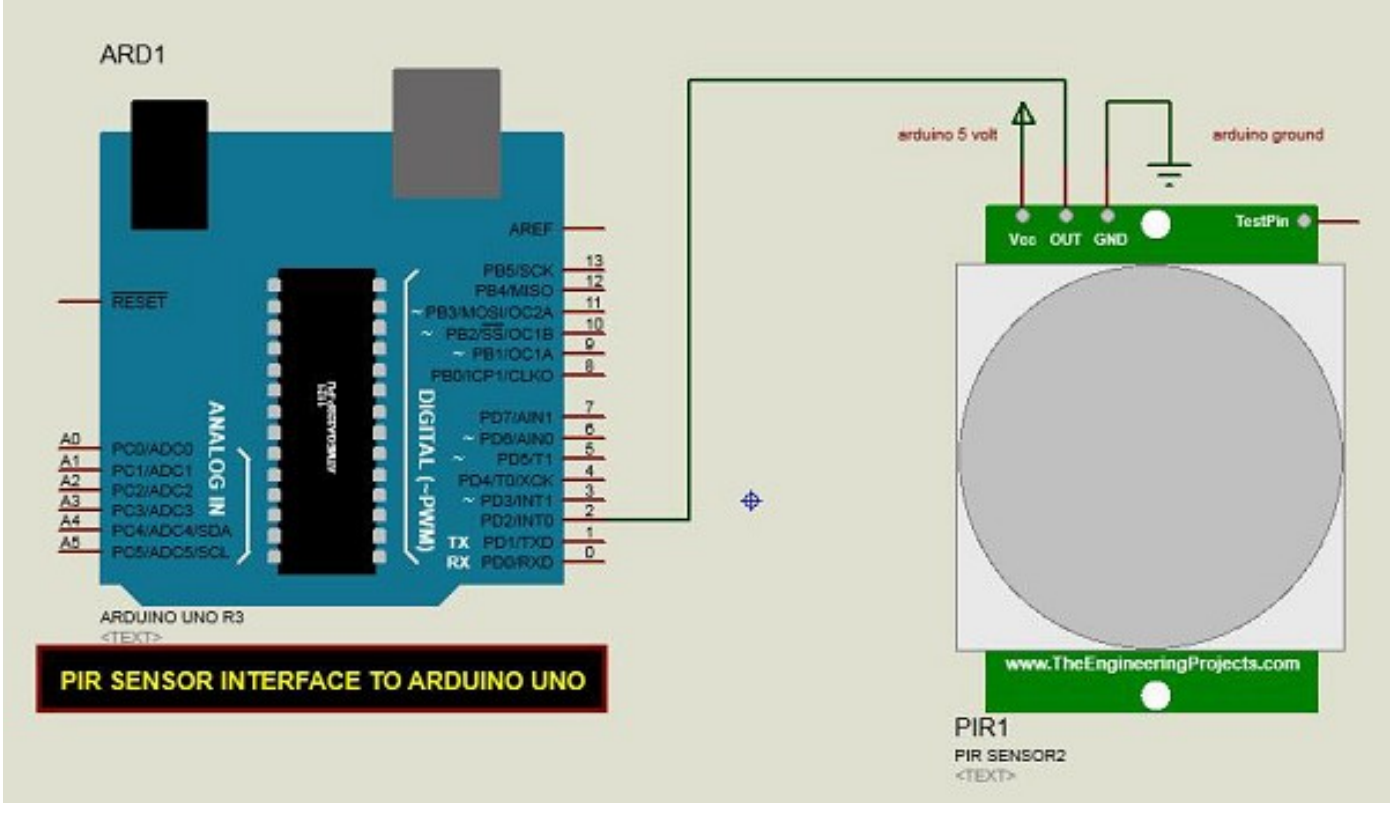


Arduino Leonardo, Micro, or Due board

# Keyboard Emulator: code

```
/*
   Keyboard Message test For the Arduino Leonardo and Micro,
      Sends a text string when a button is pressed.
   The circuit:
   * pushbutton attached from pin 4 to +5V
   * 10-kilohm resistor attached from pin 4 to ground
*/

#include "Keyboard.h"
const int buttonPin = 4; // input pin for pushbutton
int previousButtonState = HIGH; // for checking the state of a pushButton
int counter = 0; // button push counter

void setup() {
   pinMode(buttonPin, INPUT); // make the pushButton pin an input:
   Keyboard.begin(); // initialize control over the keyboard:
}

void loop() {
   int buttonState = digitalRead(buttonPin); // read the pushbutton:
   if ((buttonState != previousButtonState)&& (buttonState == HIGH)) // and
      // increment the button counter
      counter++;
      // type out a message
      Keyboard.print("You pressed the button ");
      Keyboard.print(counter);
      Keyboard.println(" times.");
   }
   // save the current button state for comparison next time:
   previousButtonState = buttonState;
}
```

# PIR Sensor: circuit

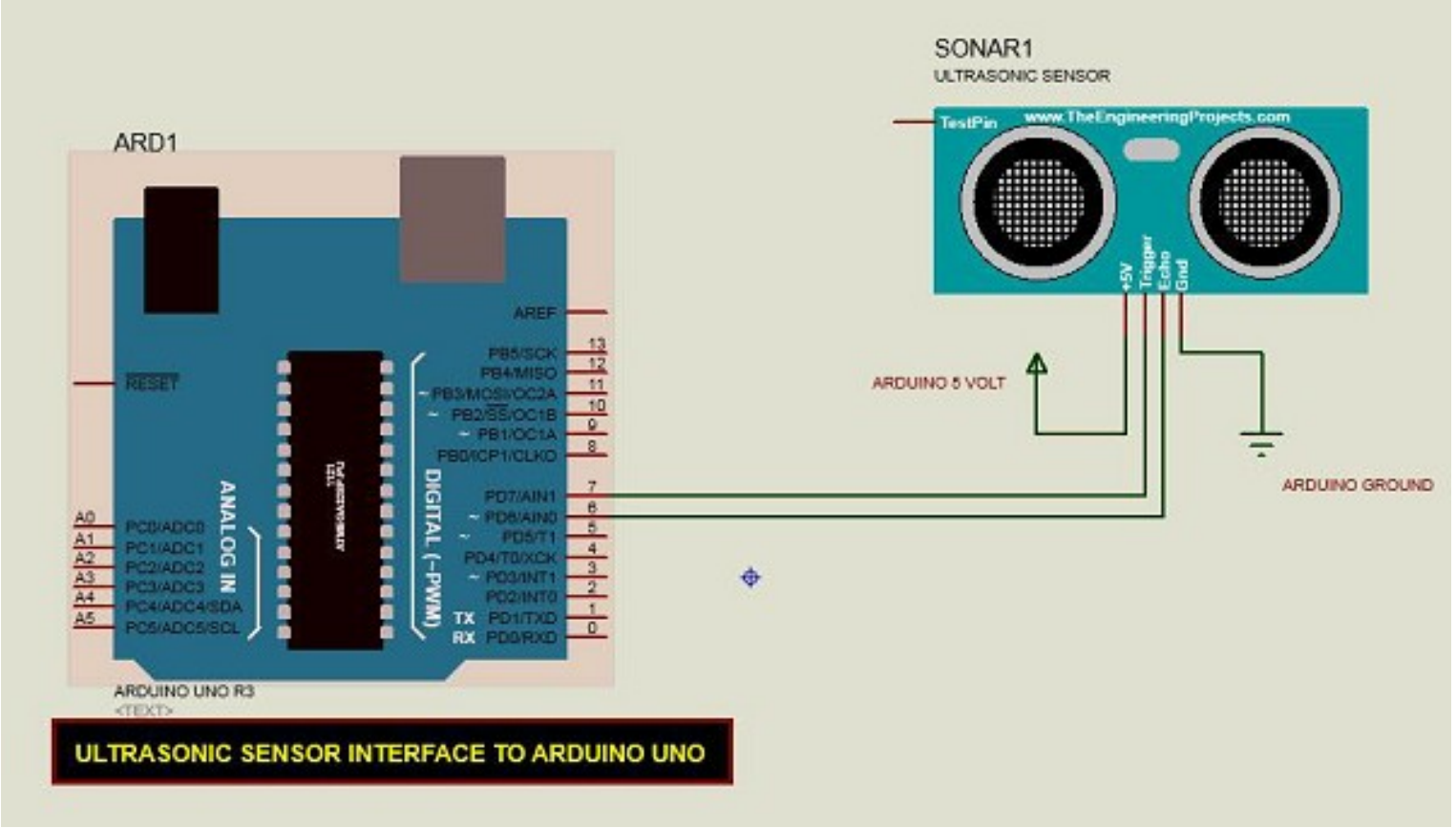



# PIR Sensor: code

```
#define pirPin 2
int calibrationTime = 30;
long unsigned int lowIn;
long unsigned int pause = 5000;
boolean lockLow = true;
boolean takeLowTime;
int PIRValue = 0;

void setup() {
   Serial.begin(9600);
   pinMode(pirPin, INPUT);
}

void loop() {
   PIRSensor();
}
```

```
void PIRSensor() {
   if(digitalRead(pirPin) == HIGH) {
      if(lockLow) {
         PIRValue = 1;
         lockLow = false;
         Serial.println("Motion detected.");
         delay(50);
      }
      takeLowTime = true;
   }
   if(digitalRead(pirPin) == LOW) {
      if(takeLowTime){
         lowIn = millis();takeLowTime = false;
      }
      if(!lockLow && millis() - lowIn > pause) {
         PIRValue = 0;
         lockLow = true;
         Serial.println("Motion ended.");
         delay(50);
      }
   }
}
```

# Ultrasonic Distance: circuit

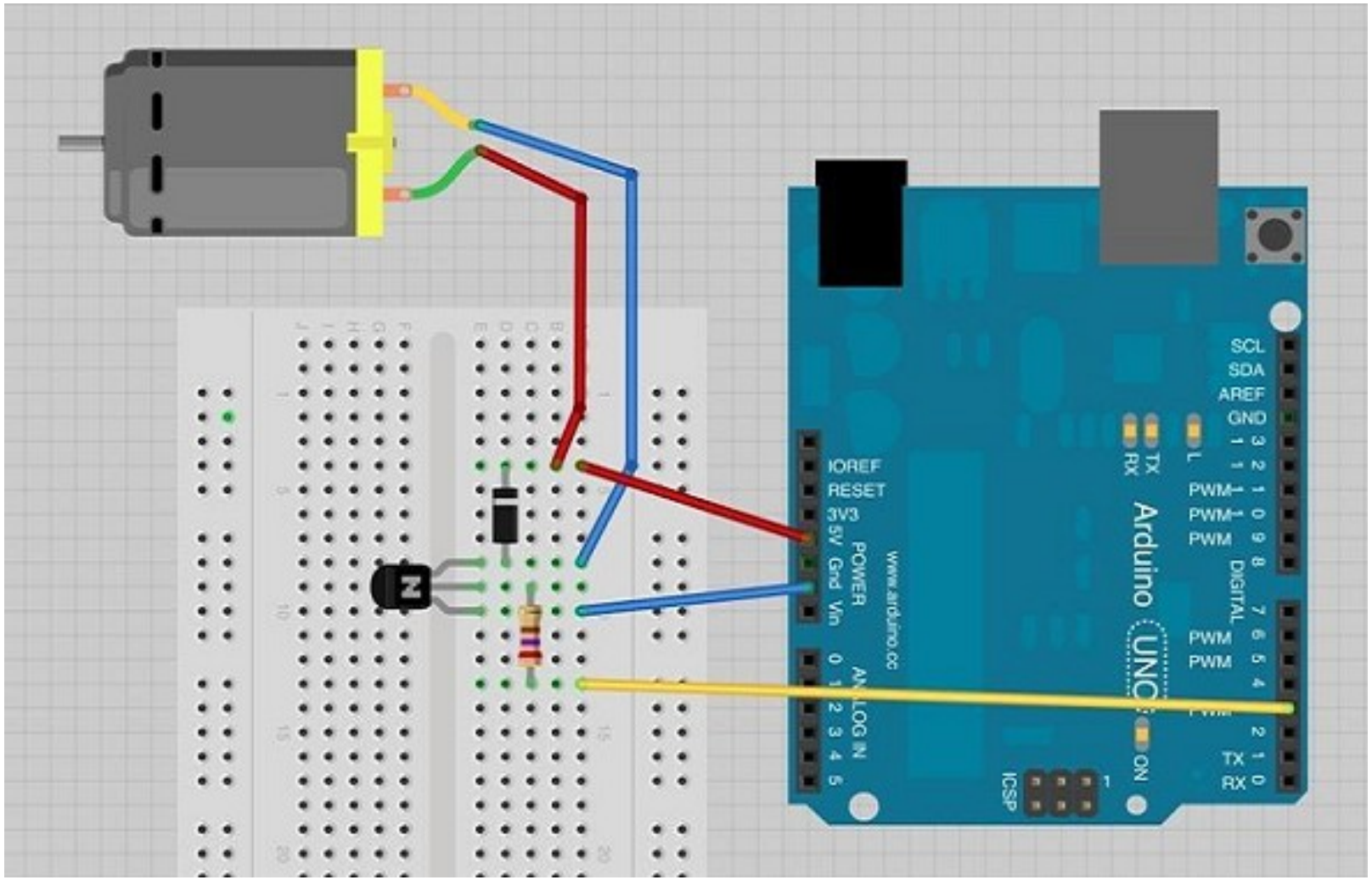



# Ultrasonic Distance: code

```
const int pingPin = 7; // Trigger Pin of Ultrasonic Sensor
const int echoPin = 6; // Echo Pin of Ultrasonic Sensor

void setup() {
   Serial.begin(9600); // Starting Serial Terminal
}

long microsecondsToInches(long microseconds) {
   return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
   return microseconds / 29 / 2;
}
```

```
void loop() {
   long duration, inches, cm;
   pinMode(pingPin, OUTPUT);
   digitalWrite(pingPin, LOW);
   delayMicroseconds(2);
   digitalWrite(pingPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(pingPin, LOW);
   pinMode(echoPin, INPUT);
   duration = pulseIn(echoPin, HIGH);
   inches = microsecondsToInches(duration);
   cm = microsecondsToCentimeters(duration);
   Serial.print(inches);
   Serial.print("in, ");
   Serial.print(cm);
   Serial.print("cm");
   Serial.println();
   delay(100);
}
```

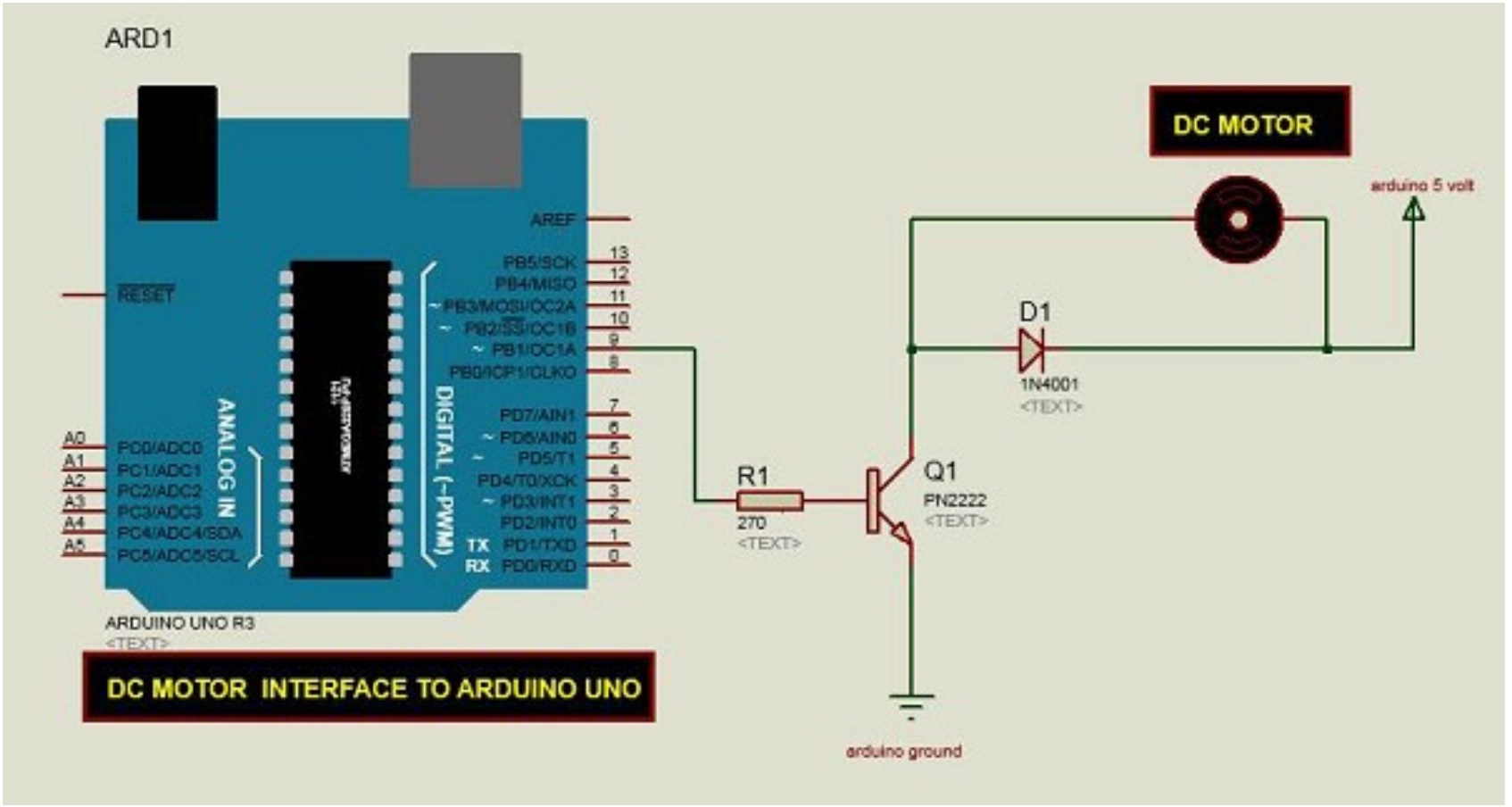
# DC Motor Spin: circuit



# DC Motor Spin: code

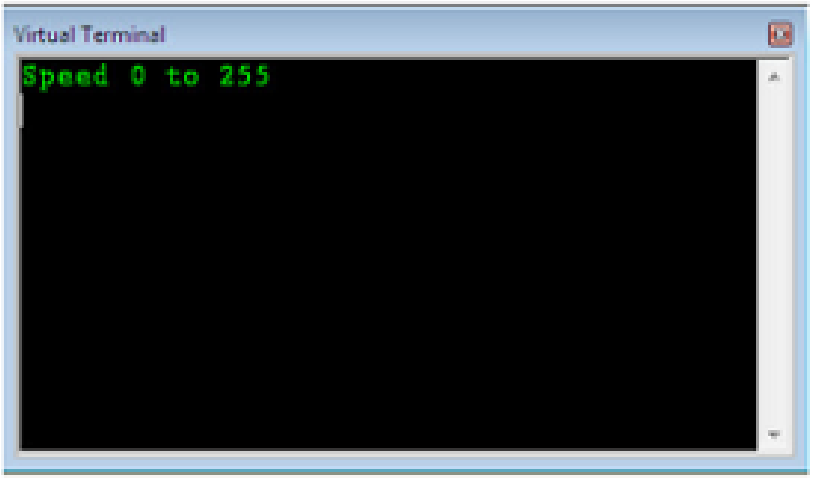
```
int motorPin = 3;

void setup() {

}

void loop() {
   digitalWrite(motorPin, HIGH);
}
```

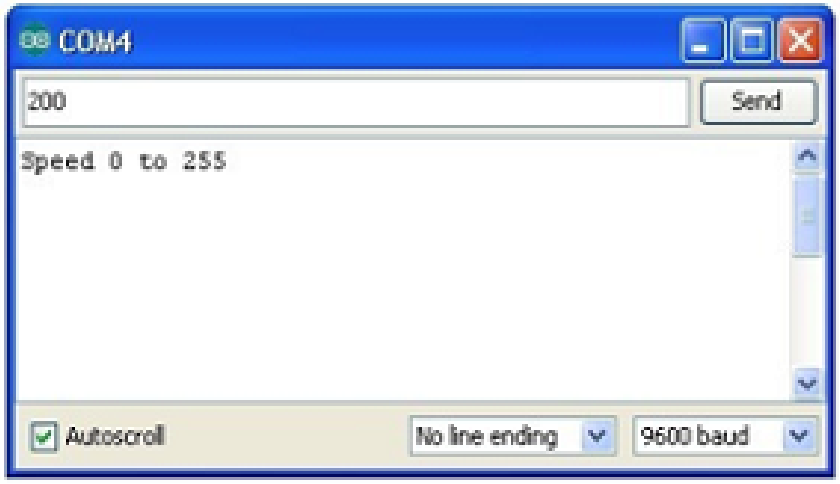
# DC Motor Speed Control: circuit



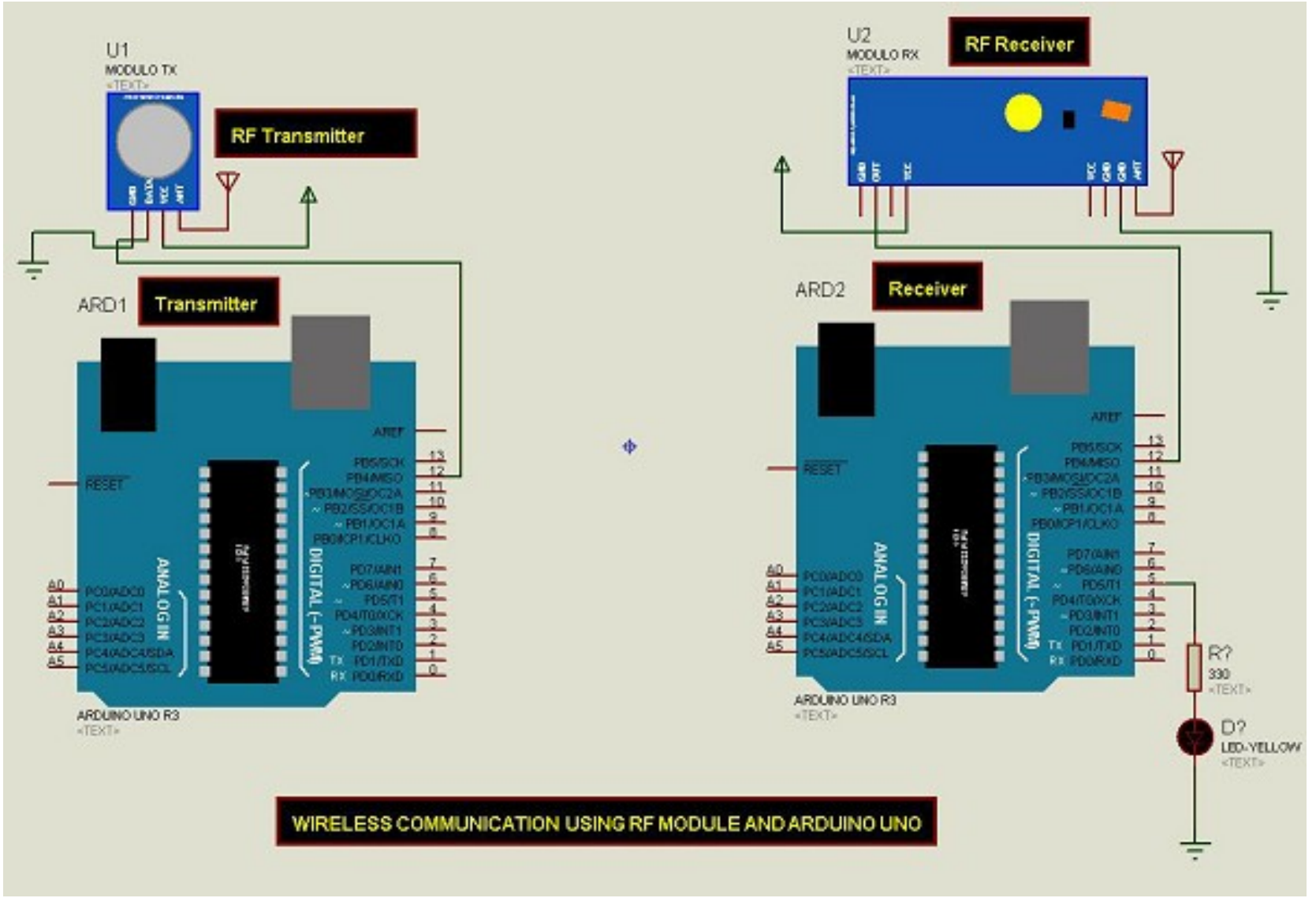
# DC Motor Speed Control: code

```
int motorPin = 9;

void setup() {
   pinMode(motorPin, OUTPUT);
   Serial.begin(9600);
   while (! Serial);
   Serial.println("Speed 0 to 255");
}

void loop() {
   if (Serial.available()) {
      int speed = Serial.parseInt();
      if (speed >= 0 && speed <= 255) {
         analogWrite(motorPin, speed);
      }
   }
}
```



# RF Communication: circuit only



# WI-FI: circuit only