

Arduino Programming

Part 3

EAS 199A
Fall 2011

Overview

Part I

- ❖ Circuits and code to control the speed of a small DC motor.
- ❖ Use potentiometer for dynamic user input.
- ❖ Use PWM output from Arduino to control a transistor.
- ❖ Transistor acts as variable voltage switch for the DC motor.

Part II

- ❖ Consolidate code into reusable functions.
- ❖ One function maps 10-bit analog input to 8-bit PWM output.
- ❖ Another function controls the motor speed.
- ❖ Using functions provides modular features that are useful for more complex control tasks, e.g. the desktop fan project.

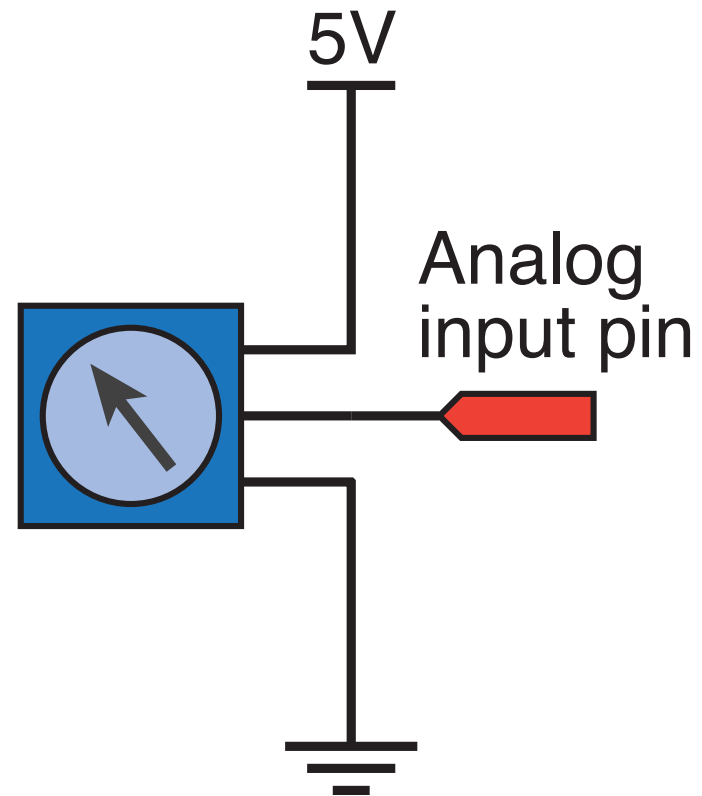
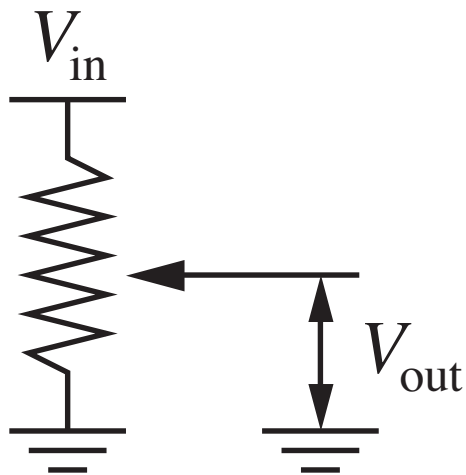
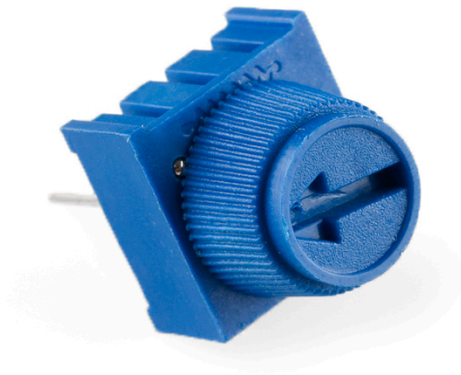
Part I: Control motor speed with a pot

Increase complexity gradually

1. Use a pot to generate a voltage signal
 - (a) Read voltage with analog input
 - (b) Print voltage to serial monitor to verify
2. Convert 10-bit voltage scale to 8-bit PWM scale
 - (a) Voltage input is in range 0 to 1023
 - (b) PWM output needs to be in the range 0 to 255
 - (c) Print voltage to serial monitor to verify
3. Write PWM data to DC motor
4. Write a function to linearly scale the data
5. Write a function to update the motor

Potentiometer Circuit

Use the potentiometer from the Arduino Inventor's Kit



Code to print potentiometer reading

```
// Function:  read_potentiometer
//
//  Read a potentiometer and print the reading

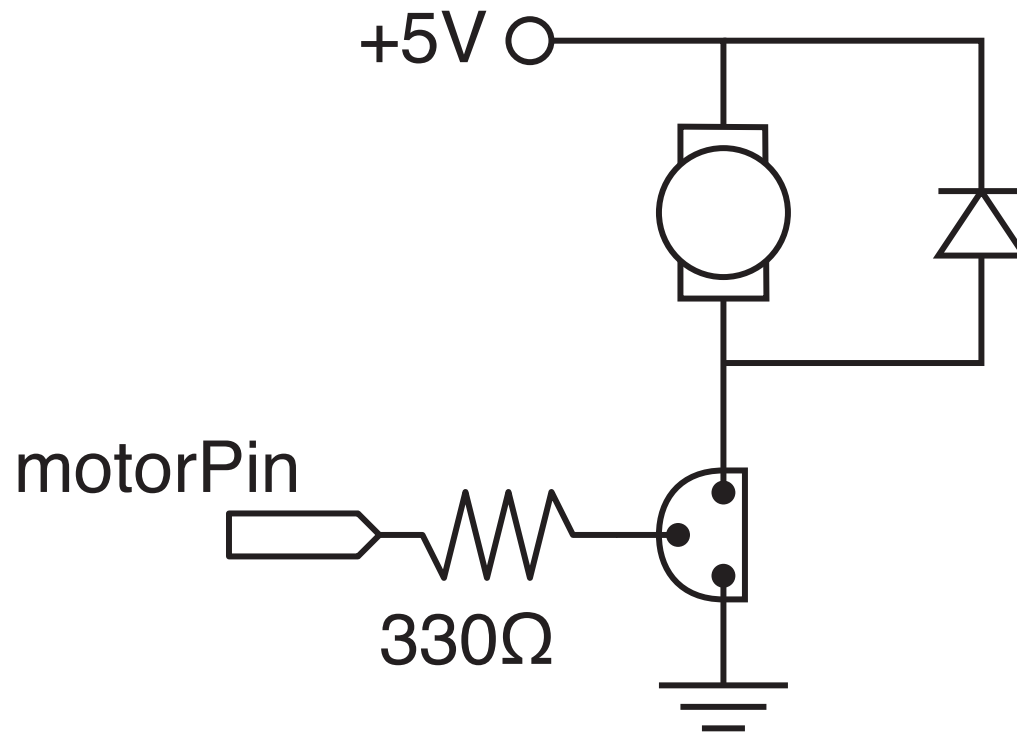
int  sensor_pin = 3;    // Wire sweeper of pot to
                        //  analog input pin 3

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

void loop()
{
  int  val;

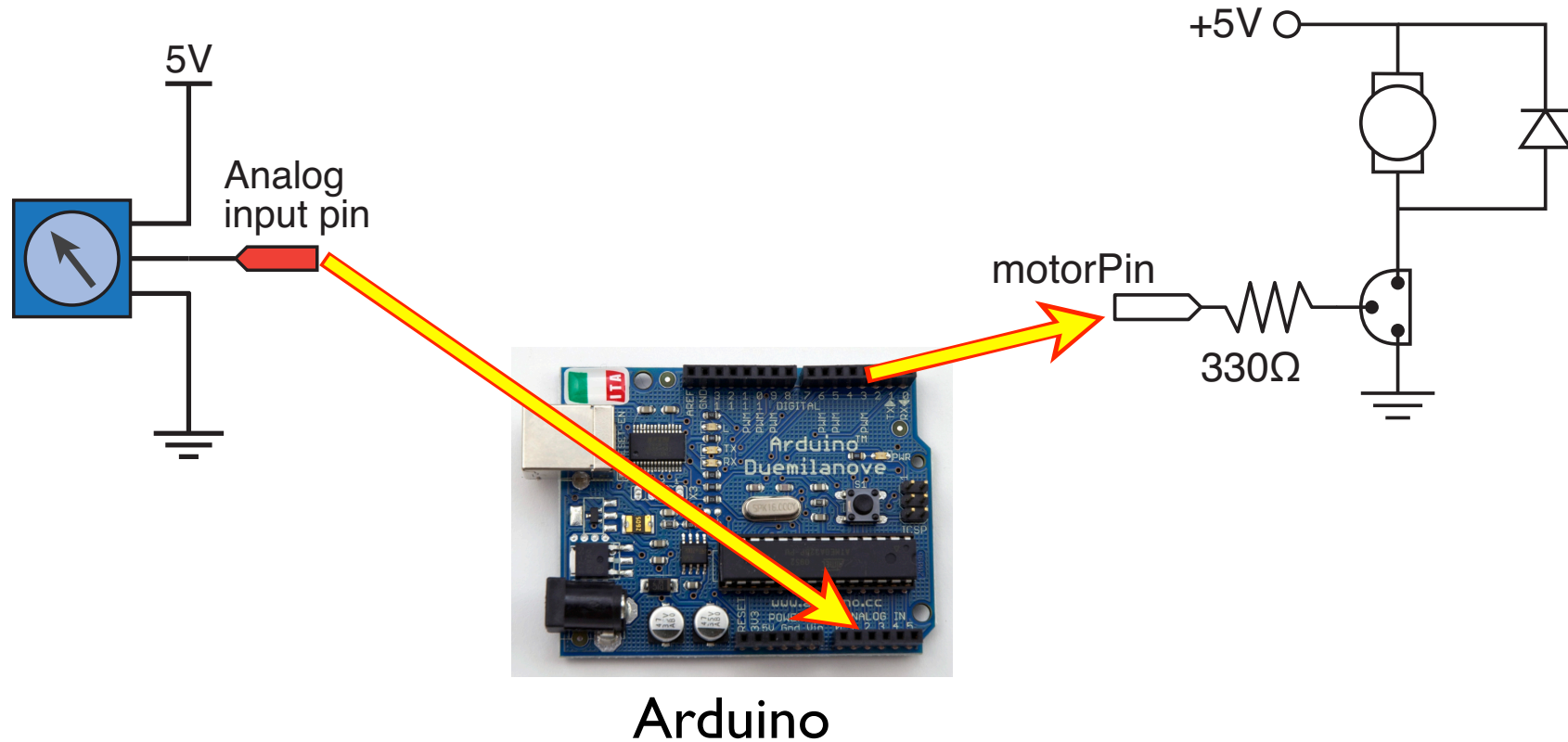
  val = analogRead( sensor_pin );
  Serial.print("reading = ");
  Serial.println( val );
}
```

DC Motor Control Circuit



Add this to the breadboard with the potentiometer circuit

DC Motor Control Circuit



Control the DC motor with PWM Output

```
// Function:  DC_motor_control_pot
//
// Use a potentiometer to control a DC motor

int  sensor_pin = 3;
int  motor_pin = 5;          // must be a PWM digital output

void setup()
{
  Serial.begin(9600);
  pinMode(motor_pin,  OUTPUT)
}

void loop()
{
  int  pot_val, motor_speed;

  pot_val = analogRead( sensor_pin );
  motor_speed = pot_val*255.0/1023.0;    // Include decimal
  analogWrite( motor_pin, motor_speed);
}
```

Subtle: Don't use integer values of 255 and 1023 here. Aggressive compilers pre-compute the integer division of 255/1023 as zero.

Part II: Create functions for reusable code

```
// Function:  DC_motor_control_pot
//
// Use a potentiometer to control a DC motor

int  sensor_pin = 3;
int  motor_pin = 5;          // must be a PWM digital output

void setup()
{
  Serial.begin(9600);
  pinMode(motor_pin,  OUTPUT)
}

void loop()
{
  int  pot_val, motor_speed;

  pot_val = analogRead( sensor_pin );
  motor_speed = pot_val*255.0/1023.0; // Include decimal
  analogWrite( motor_pin, motor_speed);
}
```

Adjust motor speed

Map input values to output scale

Final version of the loop() function

```
// Function:  DC_motor_control_pot
//
// Use a potentiometer to control a DC motor

int  sensor_pin = 3;
int  motor_pin = 5;          // must be a PWM digital output

void setup()
{
  Serial.begin(9600);
  pinMode(motor_pin,  OUTPUT)
}

void loop()
{
  adjust_motor_speed( sensor_pin, motor_pin);

  ... // do other useful stuff
}
```

adjust_motor_speed takes care of the two main tasks: reading the potentiometer output and setting the PWM signal to the transistor

Using and Writing Functions

Arduino web site

- ❖ <http://www.arduino.cc/en/Reference/FunctionDeclaration>

Functions are reusable code modules:

- ❖ Functions encapsulate details of tasks into larger building blocks
- ❖ Well-written functions can be reused
- ❖ Functions can accept input (or not) and return output (or not)
- ❖ All Arduino sketches have at least two functions
 - ▶ setup: runs once to configure the system
 - ▶ loop: runs repeatedly after start-up is complete
- ❖ Users can add functions in the main sketch file, or in separate files

The setup() Function

Consider the simple blink sketch

“void” means Returns nothing

No inputs

```
// Blink.pde: Turn on an LED for one second, then  
//           off for one second. Repeat continuously.
```

```
void setup() {  
  pinMode(13, OUTPUT);  
}
```

“setup” is the name of the function

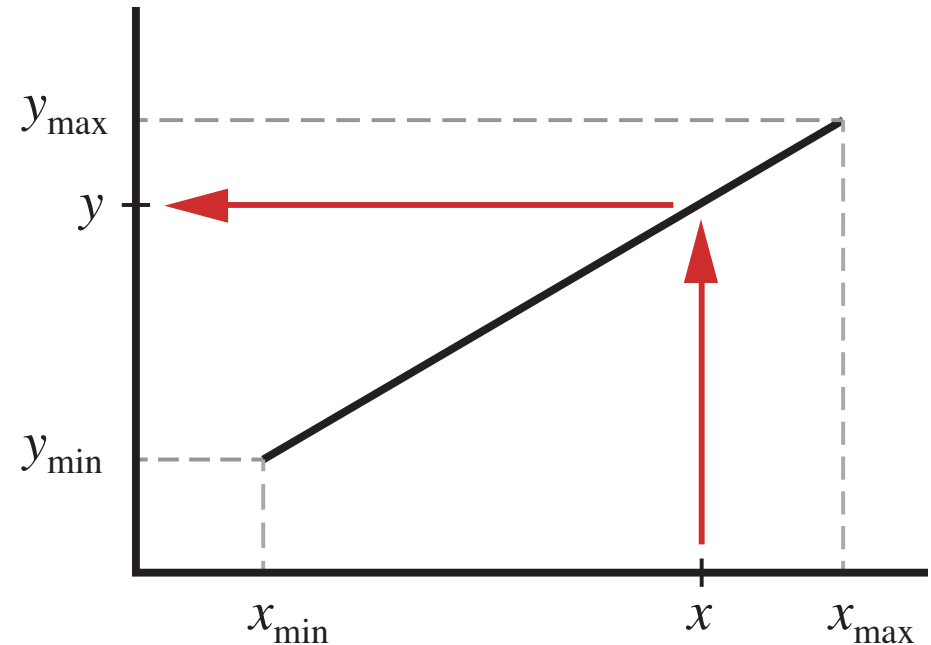
```
void loop() {  
  digitalWrite(13, HIGH); // set the LED on  
  delay(1000);            // wait for a second  
  digitalWrite(13, LOW);  // set the LED off  
  delay(1000);            // wait for a second  
}
```

A Function to Translate Linear Scales

Linear scaling from x values
to y values:

$$y = f(x)$$

where f is a linear mapping



$$\frac{y - y_{\min}}{y_{\max} - y_{\min}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

$$\implies y = y_{\min} + (y_{\max} - y_{\min}) \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

In words: Given x , x_{\min} , x_{\max} , y_{\min} , and y_{\max} , compute y

A Function to Translate Linear Scales

Enter the code at the bottom into your sketch

- ❖ The code is *not* inside any other program block (like setup or void)

How would you test that this function is working?

```
int int_scale(int x, int xmin, int xmax, int ymin, int ymax)
{
    int y;

    y = ymin + float(ymax - ymin)*float( x - xmin )/float(xmax - xmin);
    return(y);
}
```

N.B. This code is essentially a reimplementation of the built-in map function.

See <http://arduino.cc/en/Reference/Map>

A Function to Translate Linear Scales

returns an int

name is int_scale

first input is an
int named "x"

Use float for
better precision

```
int int_scale(int x, int xmin, int xmax, int ymin, int ymax)
{
    int y;

    y = ymin + float(ymax - ymin)*float( x - xmin )/float(xmax - xmin);
    return(y);
}
```

return the value stored in "y"

Functions are not nested

```
// Contents of sketch, e.g. motor_control.pde
```

```
void setup()  
{  
  ...  
}
```

```
void loop()  
{  
  ...  
}
```

```
int int_scale(int x, int xmin, int xmax, int ymin, int ymax)  
{  
  ...  
}
```



Functions call other functions

```
// Contents of sketch, e.g. motor_control.pde
```

```
void setup()  
{  
  ...  
}
```

```
void loop()  
{  
  ...  
  motor_speed = int_scale( pot_val, 0, 1023, 0, 255 );  
}
```

```
int int_scale(int x, int xmin, int xmax, int ymin, int ymax)  
{  
  ...  
  return( y );  
}
```

A diagram consisting of five red arrows pointing from the arguments of the `int_scale` function call in the `loop()` function to the corresponding parameters in the `int_scale` function definition. The arrows originate from the arguments `pot_val`, `0`, `1023`, `0`, and `255` in the call and point to the parameters `x`, `xmin`, `xmax`, `ymin`, and `ymax` respectively in the function definition.

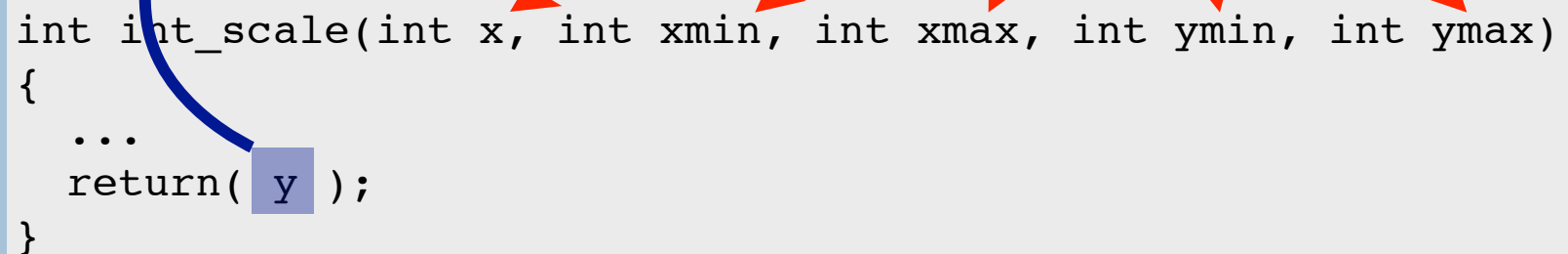
Functions call other functions

// Contents of sketch, e.g. motor_control.pde

```
void setup()  
{  
  ...  
}
```

```
void loop()  
{  
  ...  
  motor_speed = int_scale( pot_val, 0, 1024, 0, 255 );  
}
```

```
int int_scale(int x, int xmin, int xmax, int ymin, int ymax)  
{  
  ...  
  return( y );  
}
```



Use the int_scale function

```
// Function: DC_motor_control_pot
//
// Use a potentiometer to control a DC motor

int  sensor_pin = 3;
int  motor_pin = 5;          // must be a PWM digital output
```

```
void setup()
{
  Serial.begin(9600);
  pinMode(motor_pin, OUTPUT)
}
```

```
void loop()
{
  int  pot_val, motor_speed;

  pot_val = analogRead( sensor_pin );
  motor_speed = int_scale( pot_val, 0, 1023, 0, 255;
  analogWrite( motor_pin, motor_speed);
}
```

```
int int_scale(int x, int xmin, int xmax, int ymin, int ymax)
{
  int y;

  y = ymin + float(ymax - ymin)*float( x - xmin )/float(xmax - xmin);
  return(y);
}
```

A Function to update motor speed

Inputs

- ❖ sensor pin
- ❖ motor output pin

Tasks:

- ❖ Read potentiometer voltage
- ❖ Convert voltage from 10 bit to 8 bit scales
- ❖ Change motor speed

```
void adjust_motor_speed(int sensor_pin, int motor_pin)
{
    int  motor_speed, sensor_value;

    sensor_value = analogRead(sensor_pin);
    motor_speed = int_scale(sensor_value, 0, 1023, 0, 255);
    analogWrite( motor_pin, motor_speed);

    Serial.print("Pot input, motor output = ");
    Serial.print(sensor_value);
    Serial.print(" "); Serial.println(motor_speed);
}
```

Functions call functions, call functions, ...

// Contents of sketch, e.g. motor_control.pde

```
void setup()  
{  
  ...  
}
```

```
void loop()  
{  
  ...  
  adjust_motor_speed( ..., ... )  
}
```

```
void adjust_motor_speed(int sensor_pin, int motor_pin)  
{  
  ...  
  motor_speed = int_scale( ..., ..., ..., ..., ... );  
}
```

```
int int_scale(int x, int xmin, int xmax, int ymin, int ymax)  
{  
  ...  
  return( y );  
}
```