

LESSON 4

Basic Hardware Programming

The Basic Arduino Program

In this lesson, you will learn how to program the input/output ports of your STEM Board microprocessor, focusing on those functions needed for programming your robot. It is assumed that you have a basic understanding of the C programming language, as described in [Lesson 3](#). After completing this lesson, you should be familiar with the digital I/O ports, the analog-digital converter (ADC), and the pulse width modulator (PWM).

The Arduino Board

1. The Arduino UNO board and its features are shown in figure 4.1. The functionality of the Arduino UNO board is the same as your STEM Board microprocessor.

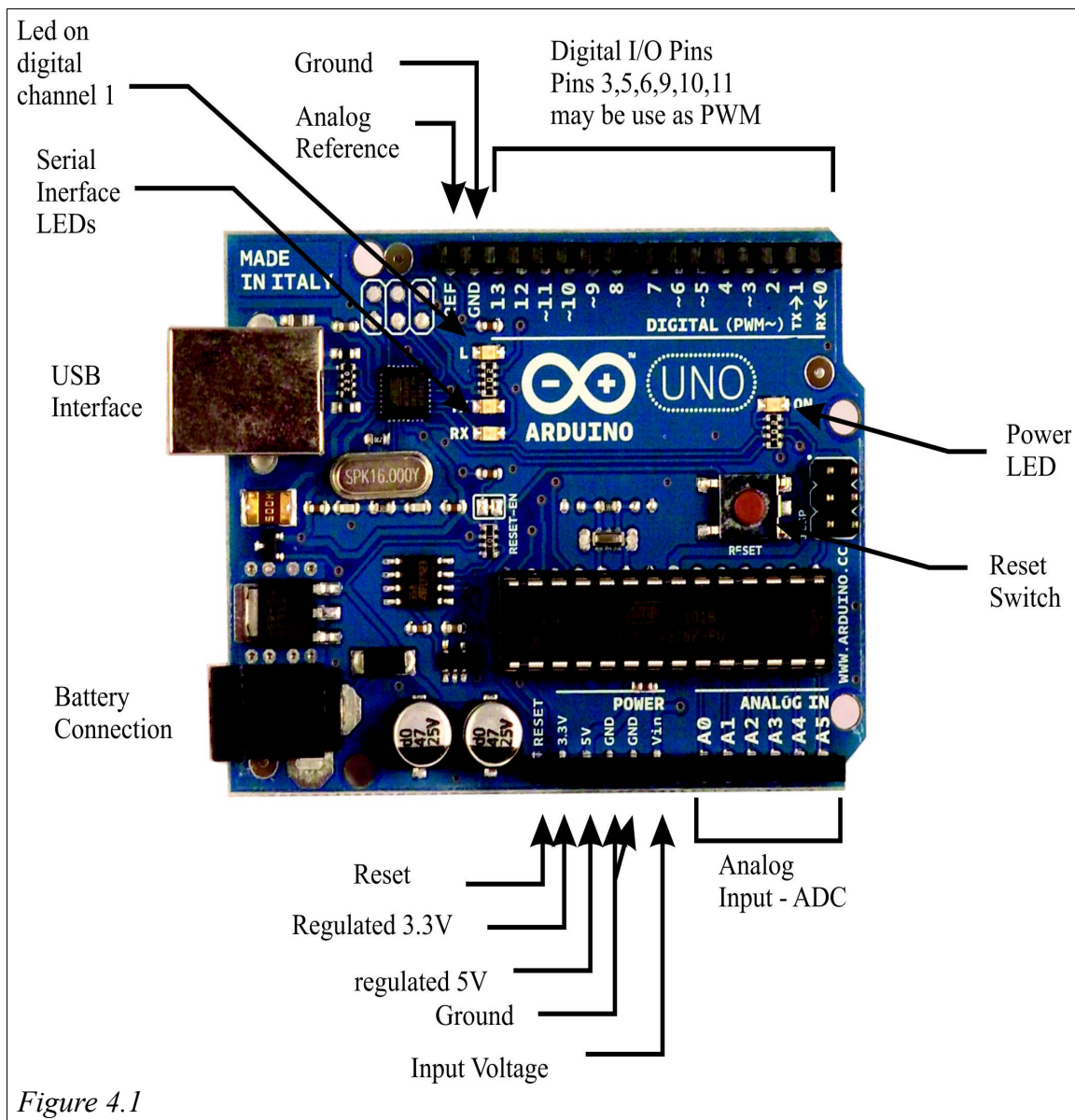


Figure 4.1

2. The specifications for the Arduino-compatible board are:

- Microcontroller ATmega328
- Operating Voltage 5V
- Input Voltage (recommended) 7-12V
- Input Voltage (limits) 6-20V
- Analog Input Pins 6
- Digital I/O Pins 14 (of which 6 provide PWM output)
- DC Current per I/O Pin 40 mA
- DC Current for 3.3V Pin 50 mA
- Flash Memory 32 KB of which 0.5 KB used by bootloader
- Analog Input Pins 6
- SRAM 2 KB
- EEPROM 1 KB
- Clock Speed 16 MHz

Digital Input/Output

1. After plugging the STEM Board microprocessor into your PC using the USB connector, you should notice that the green LED power light is turned on. You do not need to plug in the 9-volt battery at this point as the board is now powered from the USB port.
2. If everything looks good, enter and run the program Led1.pde shown in figure 4.2.

```
// Program Led 1
void setup()
{
    pinMode(13,OUTPUT);
}

void loop()
{
    digitalWrite(13,HIGH);
    delay(100);
    digitalWrite(13,LOW);
    delay(100);
}
```

Figure 4.2: Program - Led1.pde

1. As before, the program consists of two separate functions, `setup()` and `loop()`.
2. Note:
 - In the setup function, digital pin 13 is defined to an output pin.
 - Again, the loop function is the main program. The statement `digitalWrite(13,HIGH);` is a function call to the `digitalWrite`. The first parameter is the pin number, as shown at the top row of the Arduino board in figure 5.1. The second parameter, `HIGH`, is a predefined constant for 1. Likewise, `LOW` is a predefined constant for 0.

- Note that C is case-sensitive. Typing in `High` instead of `HIGH` will cause an error. That is why `HIGH` is blue and `High` is black, reminding you that you are using predefined constants. The use of predefined constants, such as `HIGH` and `LOW`, make the program much more readable and easier to debug.
- The `delay` is used to pause the program for 0.1 seconds.

Exercise 1

1. After you get the program `Led1.pde` to run, modify it to:
 - Turn on the Left LED (digital I/O pin 13) and stay lit for 1 second.
 - After the 1 second, turn off for 0.1 seconds.
 - Repeat indefinitely.
2. We can use the predefined constants to make our programs easier to read. For example, suppose pin 13 represents the left taillight for our robot. We can make a global definition that `Left_Taillight` is 13, as shown in figure 4.3.
3. Beginner programmers are hesitant to use long descriptive variables. Typing `Left_Taillight` is more time-consuming than typing 13, but program `Led2` is much easier to understand than `Led1`. They do the exact same thing. So, make your programs as easy to read as possible.

```
// Program Led2
void setup()
#define Left_Taillight 13
{
    pinMode(Left_Taillight,OUTPUT);
}

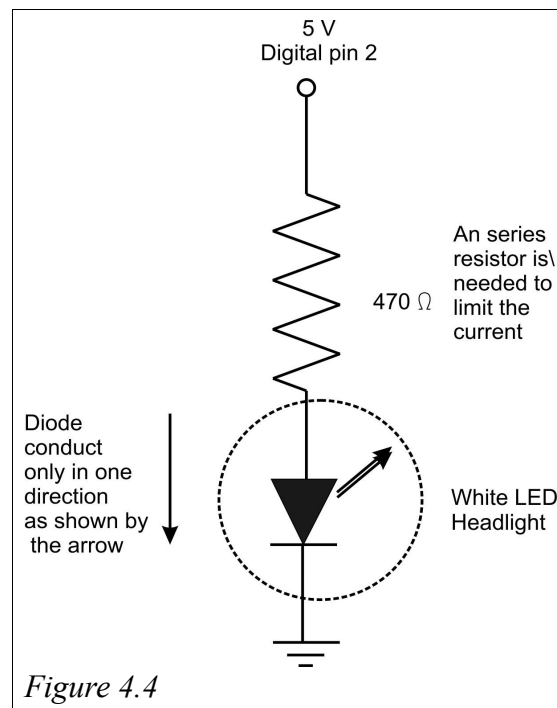
void loop()
{
    digitalWrite(Left_Taillight,HIGH);
    delay(100);
    digitalWrite(Left_Taillight,LOW);
    delay(100);
}
```

Figure 4.3: Program – Led2.pde

4. Now you understand the use of digital I/O, let's connect our own LEDs to your STEM Board microprocessor.

The Light Emitting Diode

1. LED is short for Light Emitting Diode and is shown in simple circuit in figure 4.4.



2. The LED conducts electricity only in one direction. If the polarity of the diode is reversed, it will not damage the LED, it will simply not emit light. What will damage the LED is too much current. The LEDs included in the robot kit are limited to .020 amps, commonly referred to as 20 ma (milliamps). Therefore, a limiting resistor is needed in series with the LED, as shown in figure 4.4.
3. Ohms law is used to determine the current flow through the circuit:

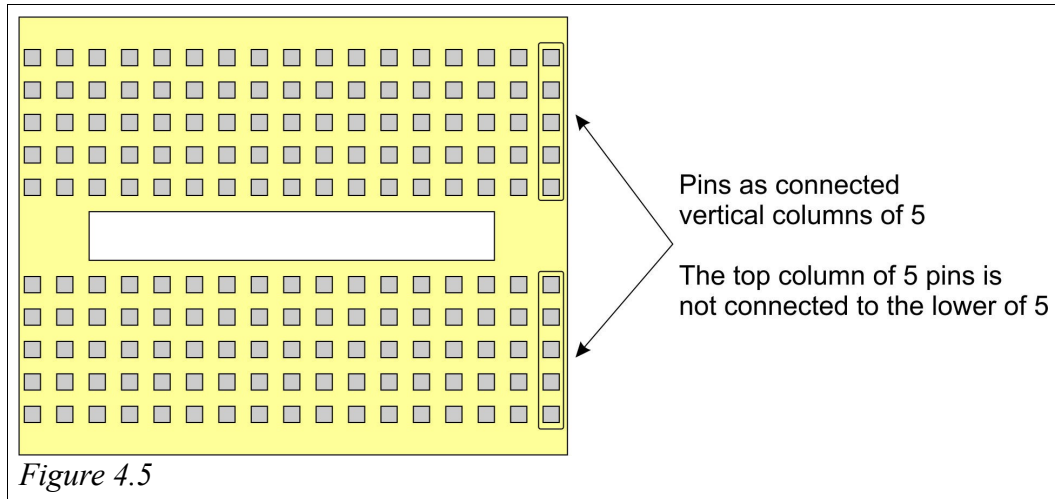
$$I = \frac{V}{R} \quad \text{or} \quad V = IR \quad \text{or} \quad R = \frac{V}{I}.$$

- V is the voltage across the component
- I is the current flowing through the component
- R is the resistance of the circuit

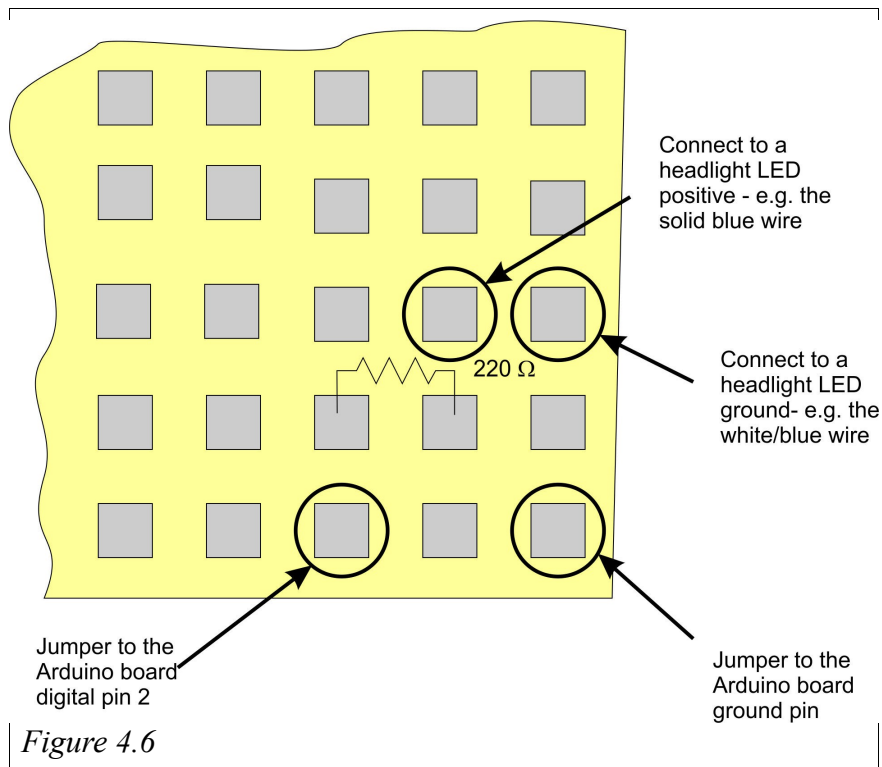
4. In circuitry, ohms is a unit of electrical resistance and is represented by the symbol Ω .
5. In our case, the LED will be connected on a digital I/O pin which has a HIGH output of 5 V. By placing a 220 Ω resistor in series, you are limiting the circuit through the resistor and LED to about 10 ma, well below the current limit of 20 ma. Let's connect the circuit shown in figure 4.4 to your STEM Board microprocessor. For this, we will need to use the breadboard.

The Protoboard

1. A protoboard, commonly referred to as a breadboard, is used for prototyping electrical circuits without needing to solder everything together.
2. A diagram of the protoboard layout is shown in figure 4.5.



3. The protoboard has 170 connections arranged in groups of five, as shown in figure 4.5. Any five connections within each group are electrically joined.
4. To connect the circuit shown in figure 4.4 to the Arduino board, see figure 4.6.



Note: The resistor provided in your kit may differ from shown ($220\ \Omega$).