*1: How does the device work?If a sensor, how is non-electrical information converted to electrical information?*

*2: What kind of interface does it have and how does that work?*

The 2 dc motors are connected using the Adafruit Motor Shield V2. In addition to providing safety and helpful libraries this shield greatly simplifies much of the circuitry we will require for driving out motors. For instance we will require the ability to drive our dc motors forwards and backwards. If we were to implement this ourselves it would require configuring pulse width modulation and a 4 transistor H-bridge or the use of a H-bridge integrated circuit such as the L293D chip. Furthermore the shield in our configuration is powered by the 5v voltage out pin on the Arduino for motor logic and directly from the 9v (6 AA batteries) power supply for the motors. To achieve a similar configuration without adding any of the safety features would require a moderate amount of additional circuitry, complexity, and additional space.

*3: What software libraries, drivers, basic code is required to use it?*

*4: If using a special library for the device, what functions are available to call?•What microcontroller platform are you using?*

Since we are using the Adafruit Motor Shield V2 we are using the software libraries supplies for interfacing with motor shield and motors. The library is included as a C++ object and supplies methods for interfacing with motors. The library is extensive and well documented available at: <https://github.com/adafruit/Adafruit_Motor_Shield_V2_Library>. The methods that are used in our program are listed as the following:

|  |  |
| --- | --- |
| **Method** | **Description** |
| **Adafruit\_MotorShield()** | the constructor used to initialize the motor shield object which the following methods are called on. |
| * **begin()** | initiates the motor shield library which initiates some default parameters as well as the Wire class. |
| * **getMotor(int pin)** | used to initialize a dc motor on a particular pin. The motor shield has its own headers for attaching motors and uses pins 1 and 2 on the Arduino board so this method is called with pins 1 and 2. This method creates and instance of Adafruit\_DCMotor which as the following methods |
| * + **run(DIRECTION)** | Sets the direction for the motor to run or releases the motor. Takes parameter FORWARD, BACKWARD, and RELEASE. |
| * + **setSpeed(int spd)** | Sets the speed of the motors as an integer from 0-255. |

*5: Is the device or microcontroller powered by 3.3v or 5v:*

*6: Any special current requirements? (may be needed for large LED displays or motors):*

*7: Names and functions of interface signals:*

The adafruit motor shield uses pulse width modulation to simulate sending varying levels of voltage as the Arduino is only capable of sending high and low voltages. This allows the motor shield to have a much higher resolution of control ( ~ .5% ). The motor shield contains all of the logic for performing motor operations such as speed and direction control and through its library provides a simple way for the Arduino programmer to control motors. Our library increases the abstraction one step further and provides the Arduino programmer with an interface for controlling the robots speed and orientation. The class we wrote includes the following methods:

|  |  |
| --- | --- |
| **Method** | **Description** |
| **Motor(int pin1, int pin2)** | The constructor method for our class which is initiated as a global variable for access in the **setup()** and **loop()** functions of our main program. This method takes two pins that are passed to the Adafruit motor shield object that establishes dc motors on the given pins. |
| * **Init()** | This method is called on the Motor object within the **setup()** function in the main program. This method is required for calling the motor shield object’s **begin()** method to establish connection with the dc motors. |
| * **drive(int speed, int direction)** | This is the main method of the Motor class for controlling the robot. This method takes a speed value and a direction value. The speed value is an int between 0-255 which again gives a ~ 0.5% resolution of speed. This method uses the **run(DIRECTION)** and **setSpeed(int spd)** from the motor shield object.  The direction value is an int representing a turning angle 0-360 degrees relative from the current position. The robot first stops if it is moving, then it turns the desired degrees on spot, then finally it will start moving this direction as the desired speed. So for example the following **drive(200, 90)** function would cause the robot to turn 90 degrees to the right, then travel at 80% speed.  The main program repeatedly calls this method with 20 degree increments away from the detected obstacle. |

*8: Demo code (commented). You must include code used to test each device independently of the others, as well as the top-level code used to demo the integration of the devices in your final team project. You must have verified the operation of the individual devices of your project before including them in your integrated team project. Omitting this independent verification of each device is a shortcut that leads to frustration and failure when problems occur!*

*9: Difficulties encountered: this is especially important if you couldn’t get the device to work. Problems and the actions taken to solve them must be well documented. Also, ask for help from your teammates and the instructor. If you can’t get it to work but have not asked for help, that’s your failure to communicate and is inexcusable.*