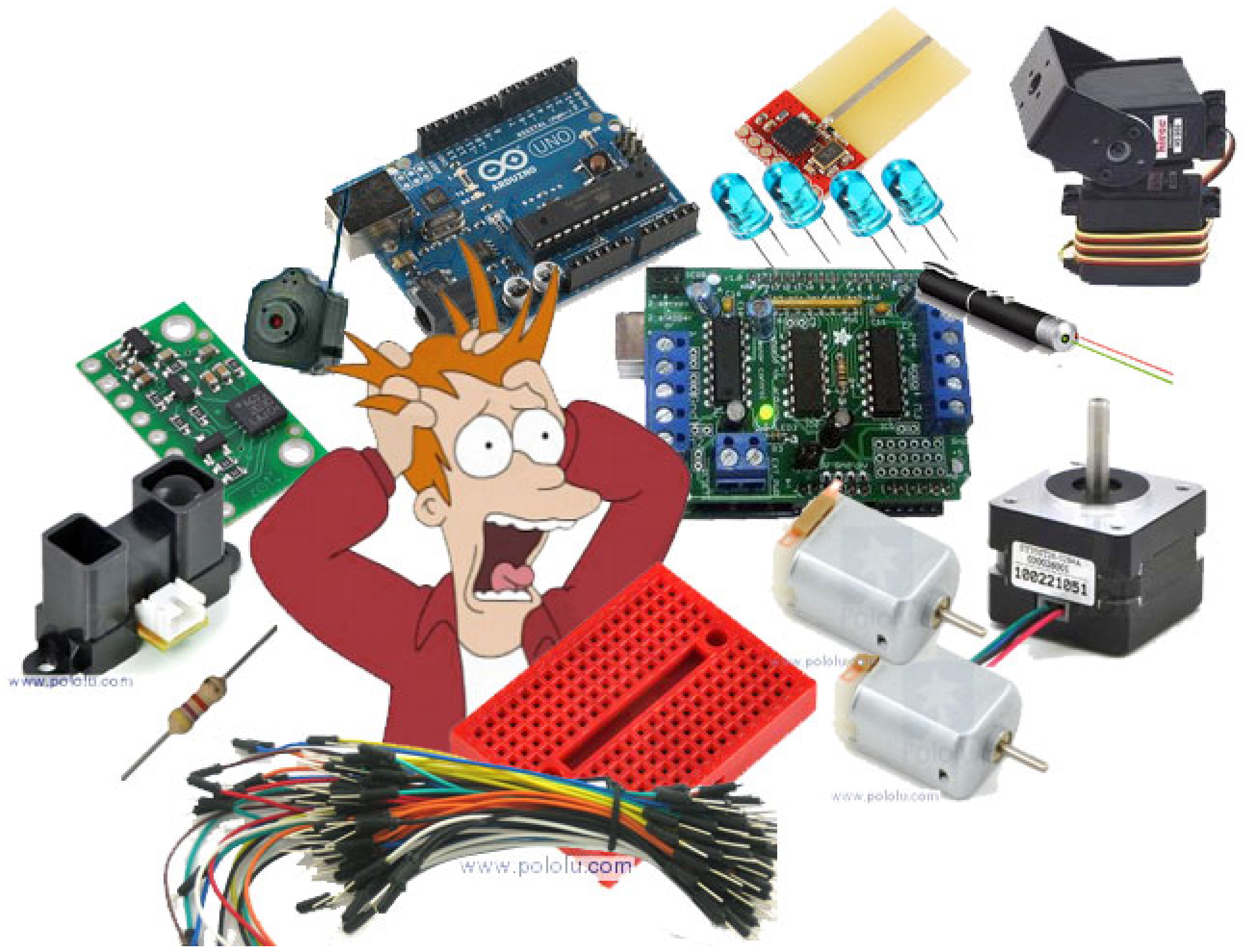
Microcontroller Based Interface Design

Part 3

System Engineering the Rover



Study of **Microprocessor Based <u>Systems</u>** and their integration with peripheral devices including sensors, actuators, and serial communications. Programming problems will be completed in C++, and at the instructor's discretion assembly, using the basic problem solving techniques learned in CECS100 and EE346. Following a progressive lab sequence, over the semester the student will design and construct a modern RISC microcontroller based system.

EE444 "Study of RISC microprocessor based systems and their integration with peripheral devices" Course Description

Introduction

Part 3 of the “System Interface Design” sequence takes you through the steps required to create the resource map used in Part 1, and the allocation of the sensors and actuators resources (plus a Long Range IR sensor) to those resources (Part 2).

Up to this point the Resource Map has been provided to you as if it just “Magically” appeared. When you work for a company, as a system engineer, you will be the one who will be responsible for creating the resource map (interface design) that others (the subsystems) work from. Technically, this is known as an **Interface Control Document** (ICD) and it provides direction to the subsytems (mechanical engineering, command and data handling, power, etc.) making up the system design (the rover).

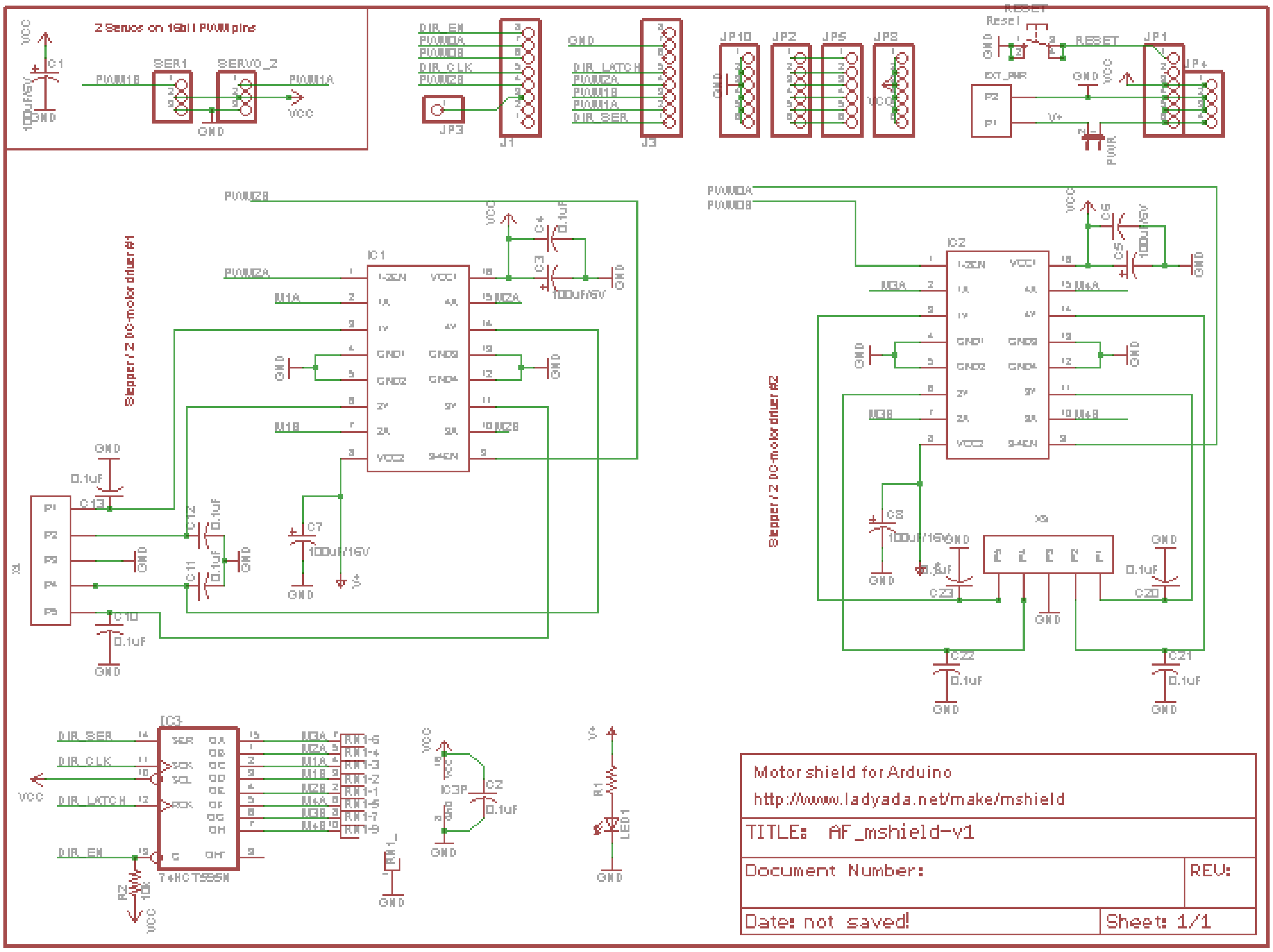
Table of Contents

1. Mission Objective

The goal of this project is to construct a robot that will be cognizant of its spatial surroundings. From this, it is to be able to circumnavigate a course comprised of 4-6 terrain features of interest placed at varying distances from the original landing site of the rover. The success of this project will be based on our rover's ability to reach these plaques (without us having to intervene and without using a design which is anything short of intelligent). Though this goal is deceivingly simple, it does, however, call for three phases of architecture that will transition our robot from an ordinary RC vehicle to a pretty sophisticated project. This document provides the detailed steps necessary to take our platform to this level. For more details on the requirements of our rover, refer to the Mission Plan.

As mentioned earlier, we will be utilizing a combination of sensors; this will gift our rover with "robo-vision". This "robo-vision" is nothing else but the employment of (2) IR Sensors in conjunction with a digital gyro. These components will be integrated so as to give our rover the ability to determine where its location is with respect to the plaques surrounding it. By this, it will, without problem, reach these goals.
(let's hope)

2. System Design



2 Servos on 16bit PWM pins
Stepper / 2 DC-motor driver #1
Stepper / 2 DC-motor driver #2
IC1
IC2
IC3
74HCT595N
Motor shield for Arduino
http://www.ladyada.net/make/mshield
TITLE: AF_mshield-v1
Document Number:
REV:
Date: not saved!
Sheet: 1/1

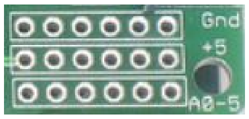
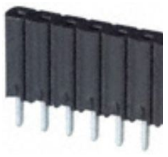
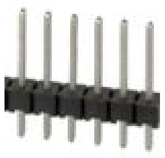
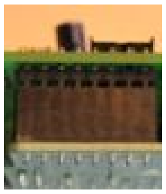
Figure 2.0 Adafruit Motor Shield Schematic

2.1 Mapping of Microcontroller Resources

One of the responsibilities of a system engineer is the allocation of resources. For a spacecraft these limited resources include mass and power. One of the limited resources on our rover is the number of I/O pins available on the ATmega328P microcontroller. Tables 2.1 and 2.2 map these I/O pins to all sensors, actuators, and associated interface electronic modules signals required to build the reference design. You can visually see when an I/O pin is allocated when the row color changes from light orange to white. In other words, once a row turns white it has been consumed. In the following section we will look at how each column was created and how our I/O pin resources have been allocated.

Terminology (A Pictorial)

If you browse the literature on how to describe the seemingly simple concept of connecting (interconnecting?) two electrical wires together you will come across a veritable tower of babel of terminology. You will hear terms like connector, socket, plug, jack, header, jumper, terminal, often prefixed or post-fixed with female or male. These terms may stand by themselves or be paired as in connector header or jumper header. For pairings order does not seem to stop anyone either as in header connector. Also hanging around these terms are words like through-hole, PCB, surface-mount, housing, DIN mount, pin, cable, ribbon, motherboard, daughter-board. For the following discussion I will try and follow the Arduino naming convention.

Through-hole solder pad  The picture above is a close-up view of a 3x6 array of through-hole pads.	Female header or Connector 	Male header or simply Header 
Jumper  An electrical connection between two points		

2.1.1 Column 1 - ATmega 328 I/O Pins

Figure 2.1 shows the ATmega328P packaged in a 28 pin Dual In-line Package (DIP). See if you can find Figure 2.1 in the ATmega328P Datasheet (hint: try page 2). Looking at each pin you will see that the majority are allocated to the three general purpose I/O ports (PB7..PB0, PC6..PC0, and PD7..PD0). All general purpose I/O pins are multiplexed with one or more ATmega328P peripheral subsystems. For example, the Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) peripheral subsystem's transmitt (TXD) and receive (RXD) interface pins share general purpose I/O pins PD1 and PD0 respectively.

If that was not enough these pins may be used to detect a change in the input state of a pin (PCINT17 and PCINT16). Known as a Pin Change INTerrupt (PCINT).

The sensors and actuators of our rover will need to be wired to our ATmega328P so we place all these I/O pins in column 1.

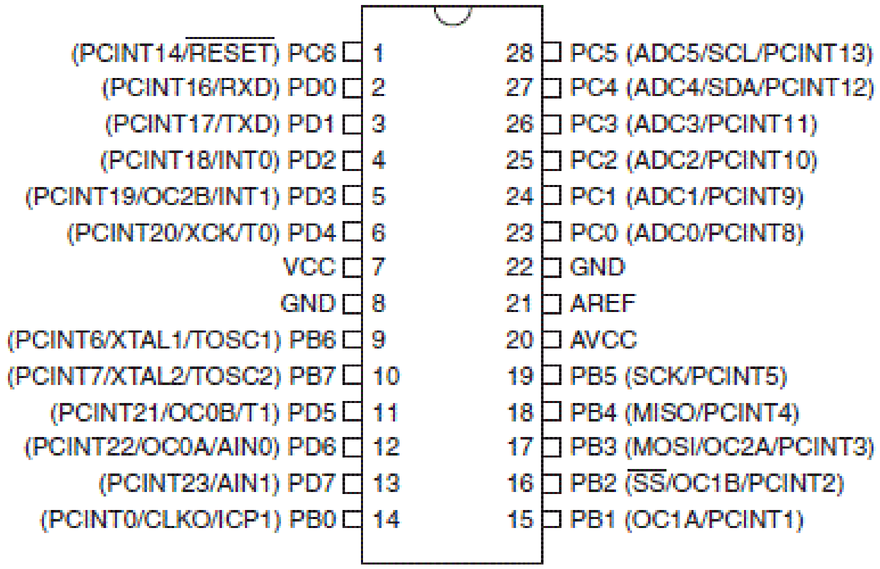


Figure 2.1 ATmega328P Pin-out

2.1.2 Column 2 - Arduino Header Pins

Our ATmega328P microcontroller is at the core of the Arduino Duemilanove. The **Arduino Duemilanove** is an open-source computing platform based on a simple low-cost I/O board and a Free Integrated Development Environment (IDE).

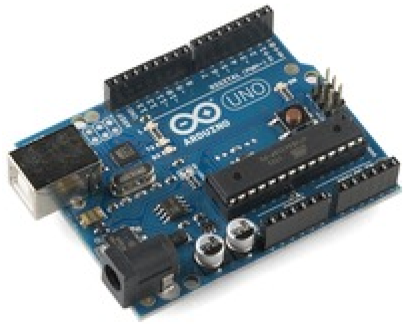


Figure 2.2 Arduino Uno

The row of connectors you see along the top and bottom edge of Figure 2.2 are wired to the I/O pins of the ATmega328P IC. The ATmega328P is located near the bottom right of the board. Unfortunately, the Arduino board uses a completely different naming convention for these I/O pins as shown in Figure 2.3. This figure is only a part of the Arduino Duemilanove Eagle Schematic, but allows us to trace the pins of the ATmega328P IC to their corresponding connectors on the Arduino. For example, our USART TXD and RXD pins are connected to header J1 pins 2 and 1 named Digital 1 and 0 (no one ever said this wasn't going to get confusing).

The second column maps these Arduino Duemilanove pins to their corresponding pins on the ATMEL ATmega328P (column 1). The following Figure is a cut-out from the Arduino Duemilanove Eagle Schematic.

Design Notes:

- J2 Pins 1 to 6 are labeled "Analog In" pins 0 to 5 on the board.
- Looking at your Arduino Duemilanove board you will see the following Digital pins labeled as PWM (Pulse Width Modulation) 3, 5, 6, 9, 10, and11.

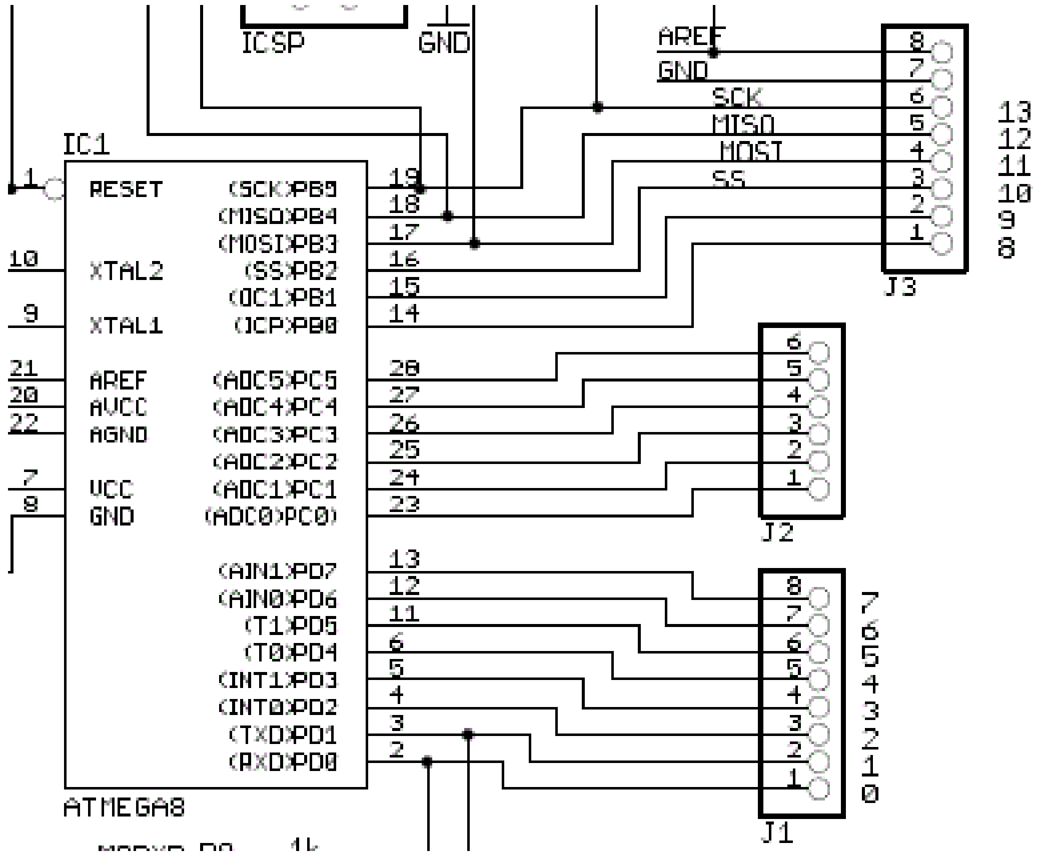


Figure 2.3 ATmega 328P Pin-out to the Arduino Duemilanove Digital and Analog Pin-out

2.1.3 Column 3 - Adafruit Motor Shield

For the reference design, we will be using the Adafruit Motor Shield. Figure 2.4 shows the Motor Shield (also known as a daughter board) plugged into the Arduino Duemilanove Board (also known as a mother board). The motor shield provides the additional logic and drivers needed to configure and power the DC motors and Bipolar stepper motor used in the reference design. In addition we will be connecting our IR sensors and digital gyro to the motor shield.

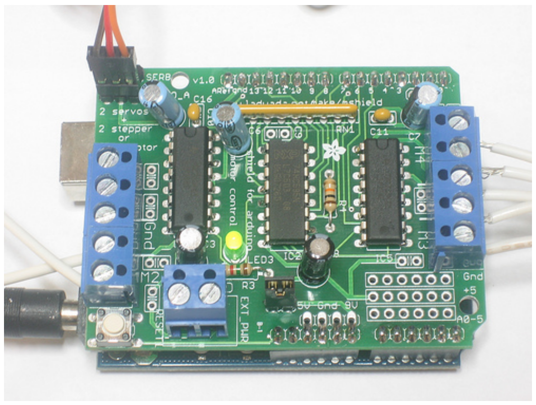


Figure 2.4 Adafruit Motor Shield

You can find the Eagle CAD PCB layout and schematic files at the links provided. If you want to open these files on your laptop you will need to download Eagle CAD. We will be using the FREE version. For reference purposes I have include the schematic of the Adafruit motor shield in Figure 2.0.

Table 2.1 Column 3 places in tabular form the Adafruit motor shield names assigned to the pins of the Atmega328P and Arduino Duemilanove (wouldn't it have been nice if they had simply all used the same naming convention). These include signal wired to Integrated Circuits IC on the the motor shield and signals wired to headers available for our sensors.

Column 3 also includes a new series of pins (X1-P5..P2, X2-P5..P2) not directly associated with an Atmega328P or Arduino. Internally these pins are wired to 2 L293D H-Bridges - a subject for the DC and stepper motor labs. Externally these screw connectors will be used to wire our DC motors (column 6 and 7) and stepper motor (column 8).

Lets begin with ATmega328P I/O resources used by the motor shield. For the purpose of mapping these resources (pins) to the Motor Shield, I have cut-out from the Motor Shield schematic (Figure 2.5) the section showing how Arduino headers J1 and J3 (see column 2) are connected to the Digital Input Register (**DIR**) and Pulse Width Modulation (**PWM**) ICs on the Motor Shield (see columns 4 and 5). These are identified as IC1, IC2, and IC3 on the schematic.

Looking at Figure 2.5, you may have noticed that header J1 pins 2 and 1 are not connected to anything on the Motor Shield. From the previous section (Figure 2.3), we know that USART TXD and RXD pins are connected to header J1 pins 2 and 1. This is actually a good idea; because these pins are used for communications between the Arduino Duemilanove board and your laptop, and are therefore not available to the motor shield.

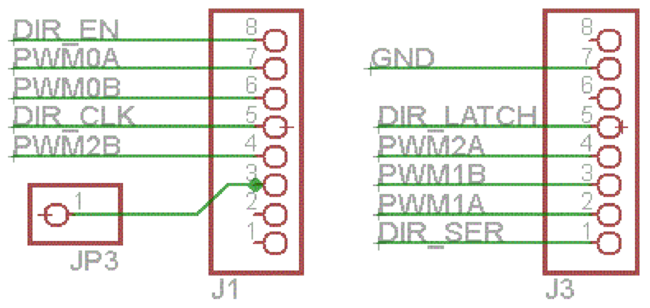


Figure 2.5 AdaFruit Motor Shield DIR and PWM wires.

2.1.4 Column 4 - Adafruit Digital Input Register (DIR)

Column 4 identifies the ATmega328P pin resources (Column 1) allocated by Adafruit to the Digital Input Register (DIR) located on the motor shield. These signals carry the DIR prefix in column 3. The DIR is implemented using a 74HC595 serial-to-parallel IC. Communications between the ATmega328P and the DIR is implemented by the Serial Peripheral Interface (SPI) protocal implemented in software. This is unfortunate because the ATmega328P includes a hardware imlemenation of the SPI, wired to a different set of pins (so we can not use it).

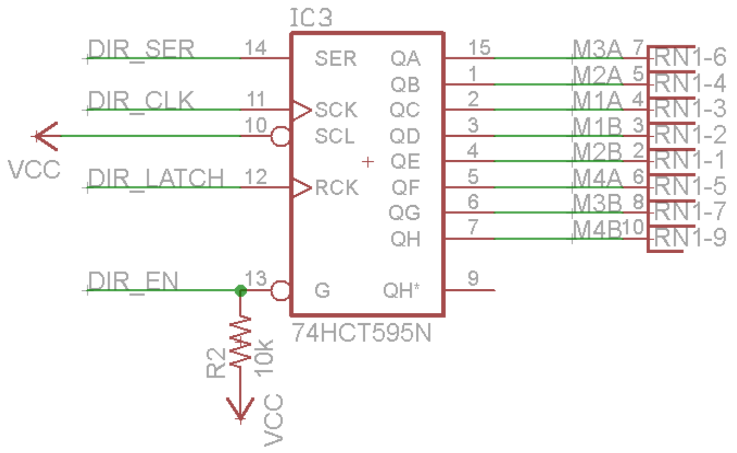


Figure 2.6 Digital Input Register (DIR) 74HCT595N

2.1.5 Column 5 - Pulse Width Modulation (PWM)

The ATmega328P includes three (3) timer subsytems (T0, T1, T2). Each timer subsystem includes two (2) compare registers (A and B). The outputs of the resulting six (6) compare circuits may be used to generate a Pulse Width Modulated (PWM) signal and are wired to I/O pins PWM0A, PWM0B, PWM1A, PWM1B, PWM2A, PWM2B. The Adafruit motor shield wires PWM0A, PWM0B, PWM2A, PWM2B to two L293D H-Bridge ICs. PWM1A and PWM1B are available for rovers that include one or two servos or as general purpose I/O pins. If the design does not use servos then you can add these pins to the list of pins available to the sensors.

2.1.5.1 PWM control of DC Motors

Figure 2.7, again cut-out from the motor shield schematic, shows how ATmega328P wires from the two compare circuits of Timer 2 (PWM2A amd PWM2B) are wired to the control inputs of one of the L293D H-Bridge ICs. With respect to Figure 2.7, configuration inputs M1A, M1B, M2A, and M2B are output from the DIR registers (Figure 2.6). You will learn more about H-Bridges in the motor labs.

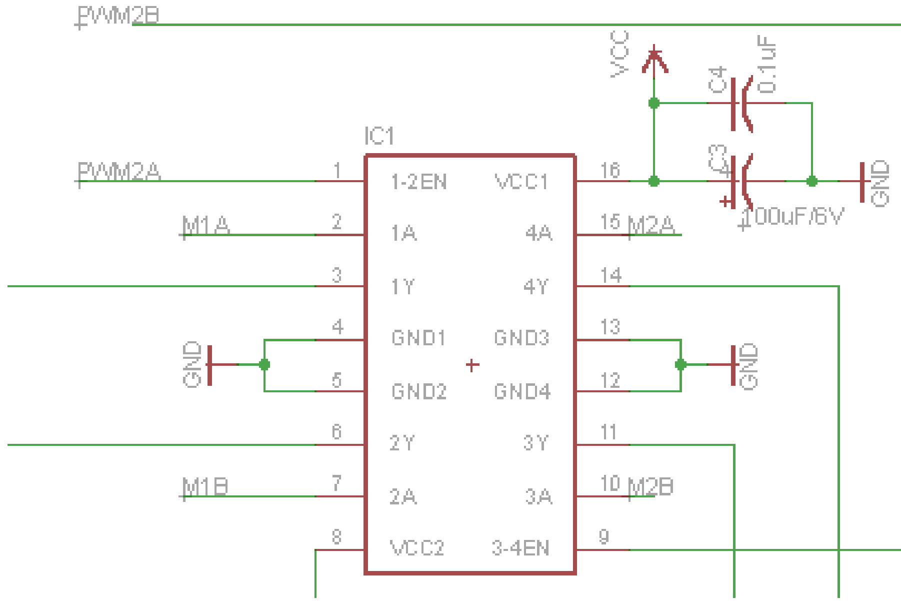
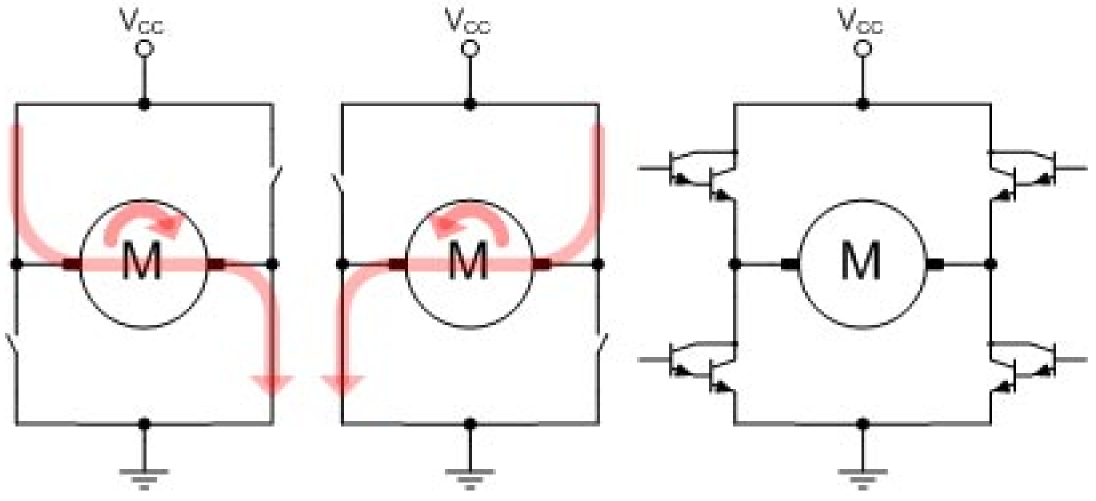


Figure 2.7 L293D Dual H-Bridge

H-Bridge

How the L293D and H-Bridges work is not the purpose of this document. However, lets take a quick look at one H-Bridge to at least conceptually see how this all comes together. For example the H-Bridge connected to the left side of the L293D IC in Figure 2.7. Consider the poles of the DC motor in Figure 2.8 wired to pins 1Y and 2Y in Figure 2.8. Each Darlington transistor pair is configured using inputs M1A and M1B from the DIR. The motor is then turned ON and OFF by the PWM2A output of the ATMega328P wired to 1-2EN.



Arduino Duemilanove Atmega 328			
1.0	H Bridge	7/11/2009	400

Figure 2.8 Conceptual Representation of a DC Motor controlled by an H-Bridge Circuit

Connectors X1 and X2 are located on the sides of your motor shields and allow you to easily connect the wires of your motors to the shield (Figure 2.4). Connector pins X1-P3 and X2-P3 go to ground. The remaining eight (8) connector pins (X1-P5,4,2,1, X2-P5,4,2,1) correspond to the outputs of the two L293D H-Bridges located on the shield. Introduced in Figure 2.7, the outputs of the two IC1 H-Bridges are wired to connector X1 as shown in Figure 2.9.

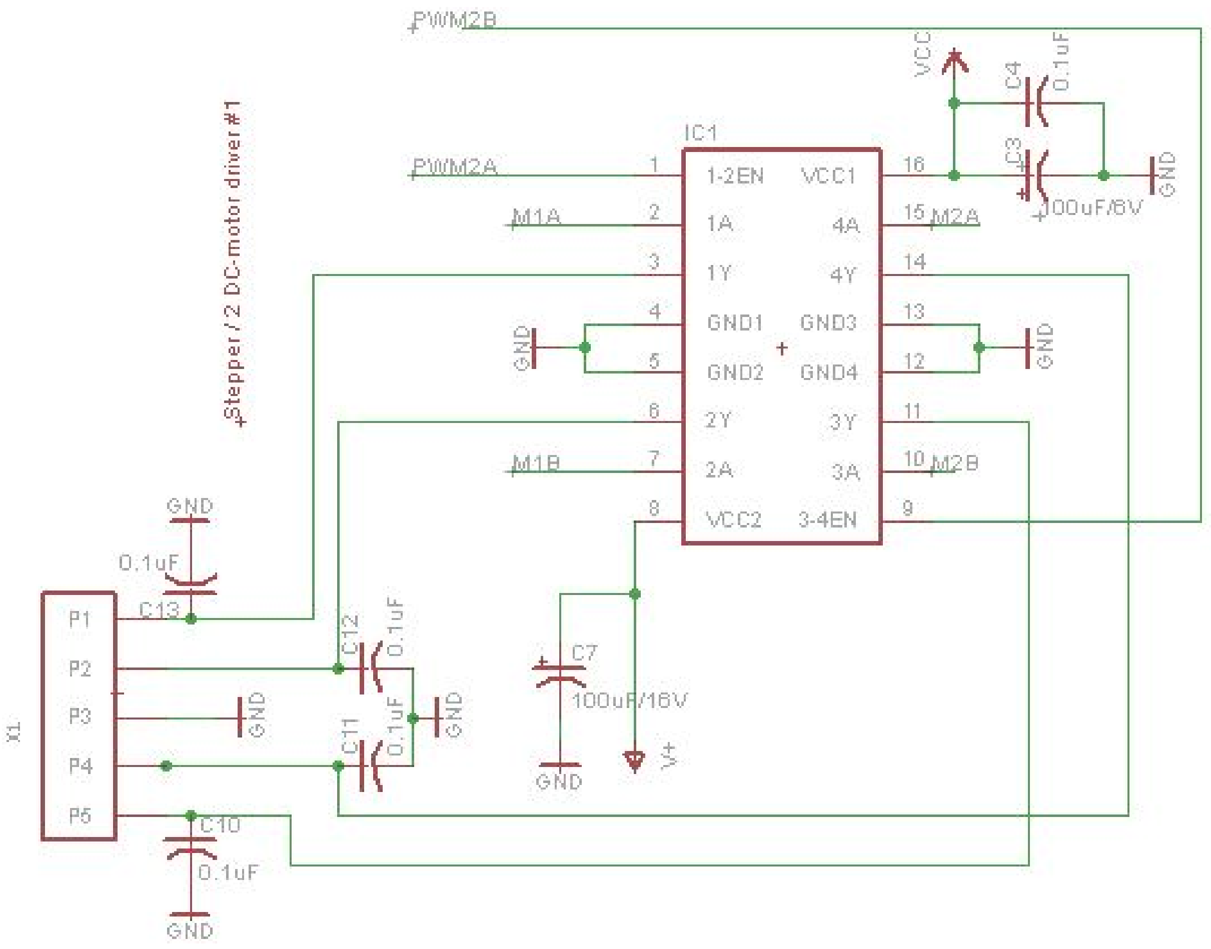


Figure 2.9 Output Connector X1

2.1.5.2 General Purpose I/O or Servo Control

Figure 2.10, again cut-out from the motor shield schematic, shows how ATmega328P pins from the two compare circuits of 16-bit Timer 1 (PWM1A amd PWM1B) are wired to the Servo jumpers located in the upper right hand corner of the motor shield (see Figure 2.4).

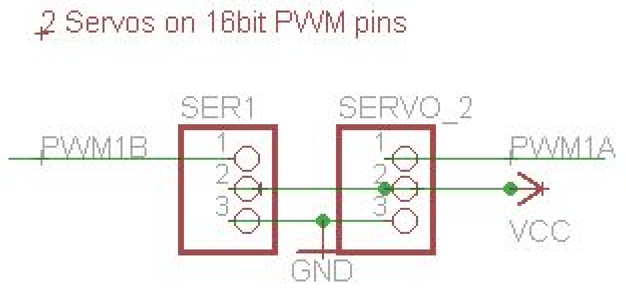


Figure 2.10 Servo Jumpers

Two 1x3 connectors can be soldered into the upper-left corner of the motor shield as viewed in Figure 2.4 “Adafruit Motor Shield.” Once soldered, the connectors will provide easy access for wiring up servos or sensors. Originally, these two 3 pin connectors were designed to connect one or two servos to the motor shield. The reference design does not include servos, in which case these jumpers provide easy access to these ATmega328P I/O pins by our sensors.

2.1.6 Column 6, 7, and 8 - Motors

You can find the outputs of connectors X1 and X2 at the bottom of Table 2.1 in Column 3 “Motor Shield.” The reference design uses two DC motors and a bipolar stepper motor. Each DC motor requires 2 wires (Forward, Reverse), while our Stepper Motor requires 4 wires. Therefore, the reference design consumes all the outputs of the 2 L293D H-Bridges. The allocation of these resources to a specific H-Bridge output is somewhat arbitrary. The ones defined in columns 6,7, and 8 are for the reference design.

Table 2.1 Microcontroller Resource Map - Motors

Atmega 328P	Arduino	Motor Shield	Motor Shield DIR (SPI Interface)	Motor Shield PWM (H-Bridge)	Left DC Motor	Right DC Motor	Bipolar Stepper Motor
PD0 (RXD)	J1-1 Digital Pin 0						
PD1 (TXD)	J1-2 Digital Pin 1						
PD2 (INT0)	J1-3 Digital Pin 2	JP3 Pin 1					
PD3 (INT1, OC2B)	J1-4 Digital Pin 3	PWM2B		IC1 Pin 9 (3-4EN)			
PD4 (T0)	J1-5 Digital Pin 4	DIR_CLK	IC3 Pin 11 (SCK)				
PD5 (T1, OC0B)	J1-6 Digital Pin 5	PWM0B		IC2 Pin 1 (1-2)EN			
PD6 (OC0A, AIN0)	J1-7 Digital Pin 6	PWM0A		IC2 Pin 9 (3-4EN)			
PD7 (AIN1)	J1-8 Digital Pin 7	DIR_EN	IC3 Pin 13 (G)				
PC0 (ADC0)	J2-1 Analog Pin 0	JP5-1					
PC1 (ADC1)	J2-2 Analog Pin 1	JP5-2					
PC2 (ADC2)	J2-3 Analog Pin 2	JP5-3					
PC3 (ADC 3)	J2-4 Analog Pin 3	JP5-4					
PC4 (ADC 4, SDA)	J2-5 Analog Pin 4	JP5-5					
PC5 (ADC 5, SCL)	J2-6 Analog Pin 5	JP5-6					

PB0 (ICP1)	J3-1 Digital Pin 8	DIR_SER	IC3 Pin 14 (SER)				
PB1 (OC1A)	J3-2 Digital Pin 9	PWM1A					
PB2 (SS, OC1A)	J3-3 Digital Pin 10	PWM1B					
PB3 (MOSI, OC2A)	J3-4 Digital Pin 11	PWM2A		IC1 Pin 1 (1-2EN)			
PB4 (MISO)	J3-5 Digital Pin 12	DIR _Latch	IC3 Pin 12 (RCK)				
PB5 (SCK)	J3-6 Digital Pin 13						
GND	J3-7 GND	GND					
AREF	J3-8 AREF						
		X1-P1 (M1)			Green		
		X1-P2 (M1)			Yellow		
		X1-P3 (GND)					
		X1-P4 (M2)				Yellow	
		X1-P5 (M2)				Green	
		X2-P1 (M4)					Yellow
		X2-P2 (M4)					Blue
		X2-P3 (GND)					
		X2-P4 (M3)					Green
		X2-P5 (M3)					Red

2.1.6 Sensors

I am going to begin this section by trying to figure out which ATmega328 I/O pins remain available for wiring up our sensors. If you get lost skip ahead to Section 2.1.6.2 which provides a summary of the GPIO pins still available for reading sensor data.

2.1.6.1 Which ATmega328 General Purpose I/O Pins are Available?

If you do not already have it running I would recommend launching Eagle CAD and opening the Adafruit “Motor shield for Arduino” schematic.

Header J1 and J3 in Figure 2.5 “Adafruit Motor Shield DIR and PWM wires” correspond to connector J1 and J3 in Figure 2.3 “ATmega 328P Pin-out to the Arduino Duemilanove Digital and Analog Pin-out.”

Lets start with connector J1. Pins 1 and 2 are reserved for communications (ATmega328 USART peripheral subsystem) between the Arduino and your PC so are not available. Connector J1 Pin 3 (Digital Pin 2) is available and wired to JP3 Pin 1 on the Motor Shield so you can easily access it for your rover.

Next lets look at Connector J3. Pin 7 provides Ground to the motor shield. Pin 8 allows us to specify an Analog Reference (AREF) voltage for Analog/Digital conversion. Connector J3-6 (Digital Pin 13) is the only I/O pin on this connector that we can access. The other I/O pins are wired to ICs on the motor shield as previously discussed. The motor shield does not provide easy access to either Pin 8 (AREF) or Pin 6 (Digital Pin 13) so you will need to solder directly to the corresponding header pin if you want to use either or use a shield stacking header

Comparing Figure 2.3 “ATmega 328P Pin-out to the Arduino Duemilanove Digital and Analog Pin-out” to the Adafruit “Motor shield for Arduino” schematic (hopefully you have it open in Eagle) you may have noticed that Arduino connector J2 is missing. This is simply an error on the Adafruit Schematic which incorrectly labels J2 as JP2.

Figure 2.11 shows this section of the motor shield schematic. All six (6) I/O pins wired to connector J2 (incorrectly labeled JP2) and named Analog Pins 0 to 5 are available. As seen in Figure 2.10, these six pins are wired directly to motor shield jumper JP5. Three 1x6 connectors can be soldered into the lower-right corner of the motor shield as viewed in Figure 2.4 “Adafruit Motor Shield.” Once soldered here, the connectors will provide easy access for wiring up our sensors.

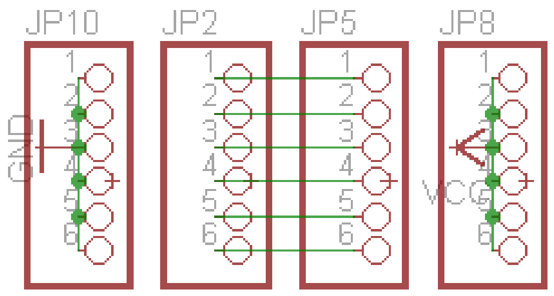


Figure 2.11 Adafruit Motor Shield Jumper J2 (incorrectly labeled JP2)

All three header (JP10, JP5, and JP8) in Figure 2.10 are conveniently located on the right-hand corner of your Motor Shield and labeled +5v, Gnd, and A0-5.

2.1.6.2 Summary of GPIO Available Resources

Table 2.2 “Microcontroller Resource Map - Sensors” provides, in a nice tabular form, everything discovered in the previous section. I have added three new columns labeled Medium IR and Gyro reflecting the sensor suite of the reference design. The rows in white in these three columns have been used up by the motor shield or motors as covered in previous sections and are not available to our sensors. Conversely, rows with a light brown background color may be used to wire up our sensors.

In summary, the following ATmega GPIO resources may be assigned to sensors.

- Digital pin 2 (J1-3) use jumper JP3 Pin 1
- Digital pin 13 (J3-6) no jumper provided
- Analog pins 0 to 5 use jumper JP5
- Digitals pin 9 (J3-2) use jumper Servo_2 - Available if you are not adding a servo. See Section 2.1.5.2 “General Purpose I/O or Servo Control” for details.
- Digital pin 10 (J3-3) use jumper Ser1- Available if you are not adding a servo. See Section 2.1.5.2 “General Purpose I/O or Servo Control” for details.

Our sensor suite does provide some additional restrictions on GPIO pin assignments, so let’s take a closer look.

2.1.7 Columns 9 and 10 - IR Sensors

The reference design uses long and medium range IR sensors for measuring the distance to an object. The output of these IR sensors is an analog voltage Vo ranging in value from a few tenths of a volt to approximately 3.1 volts. Due to the analog nature of these sensors we must wire them to two (1 per sensor) of our six (6) analog inputs. Each analog input can be selectively read by the analog-to-digital converter of the ATmega microcontroller. Which of the analog inputs we use is arbitrary.

2.1.8 Columns 11 - Gyro and I2C Interface

The gyro implements the I2C interface protocol for communications with the ATmega328P. Physically, the I2C is a serial data bus that allows multiple external devices to be connected using only 2 I/O pins on our Arduino. This data bus is typically used for devices with slower data transfer rates such as serial-enabled LCD displays or in our case, the gyro. The code for the gyro uses the Arduino’s 'wire.h' library that takes care of the I2C interfacing. Wire.h uses analog pin(4) for the serial data line (labeled SDA on the gyro schematic) and analog pin(5) for the serial clock (labeled as SCL on the gyro schematic).

Table 2.2 Microcontroller Resource Map - Sensors

Atmega 328P	Arduino	Motor Shield	Long IR	Medium IR	Gyro

PD0 (RXD)	J1-1 Digital Pin 0				
PD1 (TXD)	J1-2 Digital Pin 1				
PD2 (INT0)	J1-3 Digital Pin 2	JP3 Pin 1			
PD3 (INT1, OC2B)	J1-4 Digital Pin 3	PWM2B			
PD4 (T0)	J1-5 Digital Pin 4	DIR_CLK			
PD5 (T1, OC0B)	J1-6 Digital Pin 5	PWM0B			
PD6 (OC0A, AIN0)	J1-7 Digital Pin 6	PWM0A			
PD7 (AIN1)	J1-8 Digital Pin 7	DIR_EN			
PC0 (ADC0)	J2-1 Analog Pin 0	JP5-1			
PC1 (ADC1)	J2-2 Analog Pin 1	JP5-2	Vo		
PC2 (ADC2)	J2-3 Analog Pin 2	JP5-3		Vo	
PC3 (ADC 3)	J2-4 Analog Pin 3	JP5-4			
PC4 (ADC 4, SDA)	J2-5 Analog Pin 4	JP5-5			SDA
PC5 (ADC 5, SCL)	J2-6 Analog Pin 5	JP5-6			SCL
PB0 (ICP1)	J3-1 Digital Pin 8	DIR_SER			
PB1 (OC1A)	J3-2 Digital Pin 9	PWM1A			

PB2 (SS, OC1A)	J3-3 Digital Pin 10	PWM1B			
PB3 (MOSI, OC2A)	J3-4 Digital Pin 11	PWM2A			
PB4 (MISO)	J3-5 Digital Pin 12	DIR _Latch			
PB5 (SCK)	J3-6 Digital Pin 13				
GND	J3-7 GND	GND			
AREF	J3-8 AREF				

2.2 Mapping of Power Resources

This sections lists the power requirements for electrical components comprising the reference design. Electrical components are divided by 1) digital electronic systems and sensors; this is your "clean" power supply and 2) actuators and/or motors, this is your "dirty" power. Digital electronic systems include, the Arduino Duemilanove and Adafruit motor shield. Sensors include the (2) IR rangers and gyro. Motors include the two (2) DC motors and the bipolar stepper motor.

Powering the Motors from the Adafruit Motor Shield

Reference: How to set up the Arduino + Shield for powering motors

Figure 2.14 is cut-out from the Adafruit Motor Shield Schematic (Figure 2.0) and shows how either the connector providing power to the Arduino (Vin *not* Vcc) or an external connector on the motor shield (EXT_PWR) may be used to provide power to the motors (V+).

The toy motors used on the reference design are small enough that they may be powered directly from the 9v battery connector or USB connector on the Arduino Duemilanove. On the schematic this power source is located on JP1 pin 6 (Vin). By placing a jumper (PWR pins 1 and 2) on the motor shield you can connect this power source directly to V+. If you add this jumper you should never have anything connected to the External Power (EXT_PWR) connector. **Doing so will most probably destroy your Arduino Duemilanove and possibly damage your USB port and/or Laptop.**

To avoid accidental leaving the jumper in place when attaching your motor batteries to the the external connector (EXT_PWR) on the motor shield, I would highly recommend always powering your motors from the motor batteries and that you **never add the power (PWR) jumper**.

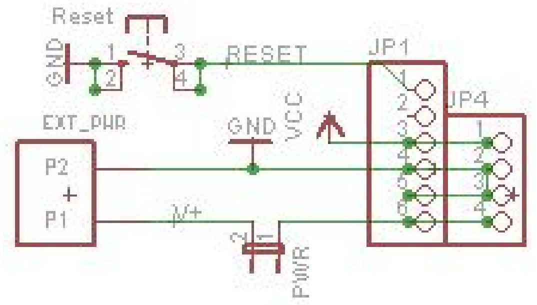


Figure 2.14 Adafruit Motor Shield Power

The safest strategy for working with electro-mechanical systems to is provide a "dirty" and a "clean" power source. The dirty supply provides power to the actuators (servos, solenoids) and DC motors. Typically, these actuators and motors involve switching on and then off inductive loads. This switching produces a lot of noise on the power supply input to the digital devices which can result in crashes (in this case literally). To solve this problem we will provide the digital systems including the sensors with there own "clean" supply of power. For the reference design we simply use a 9v battery to supply our “clean” power. I am not sure how long a standard 9v battery can provide power to our digital systems; and have therefore put together this power budget (analysis).

During the semester, you will be asked to fill-in this "system/test" level role for the project. Specifically, you will create a list of all systems drawing power (current) from the 9v battery (Arduino, Project Shield (clean side), IR sensor, PING, etc.) and then divide by the amp-hour rating of a good 9v battery taking into the efficiency of our voltage regulator. This should allow us to identify problems early and explore alternative solutions (different battery configurations) if warranted.

Later you will directly measure the current drawn using your multimeter. Once you have the current draw, you can use Simulink to model the battery and estimate the operating time of the system. Finally, you can verify this predicated value with real-world testing.

The following values were provided by the Wall-E v4.5 team and need to be updated to the reference design.

Table 2.3 "Dirty" Power - Calculated

Component	Voltage (V)	Current (A)	Power (W)	Source
Bi-Polar Stepper Motor	12	0.330	3.69	

DC Motor (x2)	6	0.21	1.26	
Total		0.54	4.95	

Table 2.4 "Clean" Power - Calculated

Component	Voltage (V)	Current (mA)	Power (mW)	Source
Long Range IR	5V	30	150	
Medium Range IR	5V	33	165	
Gyro	3.3	1	33	
Adafruit Motor Shield	12V	3	36	
Arduino Duemilanove	5V	45	225	see note 1
Total		112	576	

Notes:

1. Instructable lists a measured current of 18.7mA at 5v, while Adafruit places current at about 10mA, with the observation that you can reduce this a lot by making it go to sleep.

4 Wiring Diagrams

4.2 Hardware Wiring Quick Reference Guide

The following table maps the actuators and sensors to their immediate terminals on the Motor Shield. This is meant to be a quick reference for hardware wiring of the reference design.

Name	Sensor Mapping	Actuator Mapping

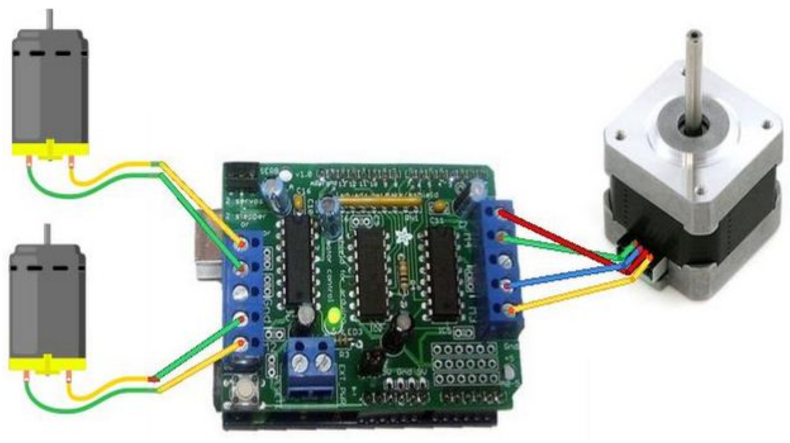
Left DC Motor Track		M1 on Motor Shield. Refer to Fritzing diagram for specific wiring.
Right DC Motor Track		M2 on Motor Shield. Refer to Fritzing diagram for specific wiring.
Bipolar Stepper Motor		M3 & M4 on motor shield. Refer to Fritzing diagram for specific wiring.
SparkFun HMC6352 gyro Module	From North arrow GND pin 1 (square pin) Vcc - 3.3 or 5 v, pin 2 SDA - data, pin 3 SCL - clock, pin 4	
Sharp IR Sensor (Long Range) GP2Y0A710YK0F	Front view "sharp", left to right, pins 1 to 5 Vo - output, pin 4 green wire) Vcc - 5v, pins 2 & 3 red wire) GND, pins 1 & 5 black wire)	
Sharp IR Sensor (Medium Range) GP2Y0A02YK	Front view "sharp", left to right, pins 1 to 3 Vo - pin 1 green wire GND - pin 2 black wire Vcc - 5v, pin 3 red wire	

4.3 Fritzing Diagrams

Here, we physically show how to connect all of the hardware components together.

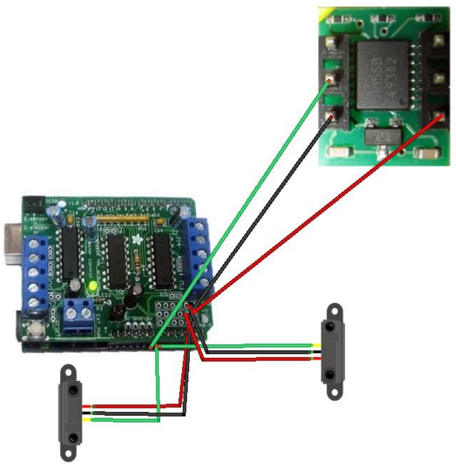
4.3.1 Motors

First, we detail all the actuators (2-DC Motors & 1-Stepper Motor)



4.3.2 Sensors

Next, we diagram all of our sensors (gyro & IRs). Because the color of wires represent different values at different levels of electrical applications, we'll quickly recap what the color of each wire terminates to; black is ground (GND); red is power (+5V); and, green is the output signal (V0).



Appendix A - Online Resources

- Adafruit Motor Shield Servos, steppers and DC motors!
- Robotshop Adafruit Motor Shield RB-Ada-02
- Understanding the Adafruit Motor Shield Library
- Control your motors with L293D and Arduino
- DC Motor Control
- ITP Physical Computing, DC Motor Control Using an H-Bridge
- L293D dual H-bridge motor driver ICs
- 74HCT595N 8-bit shift register