Microcontroller Based Interface Design

Part 2

System Engineering the Rover

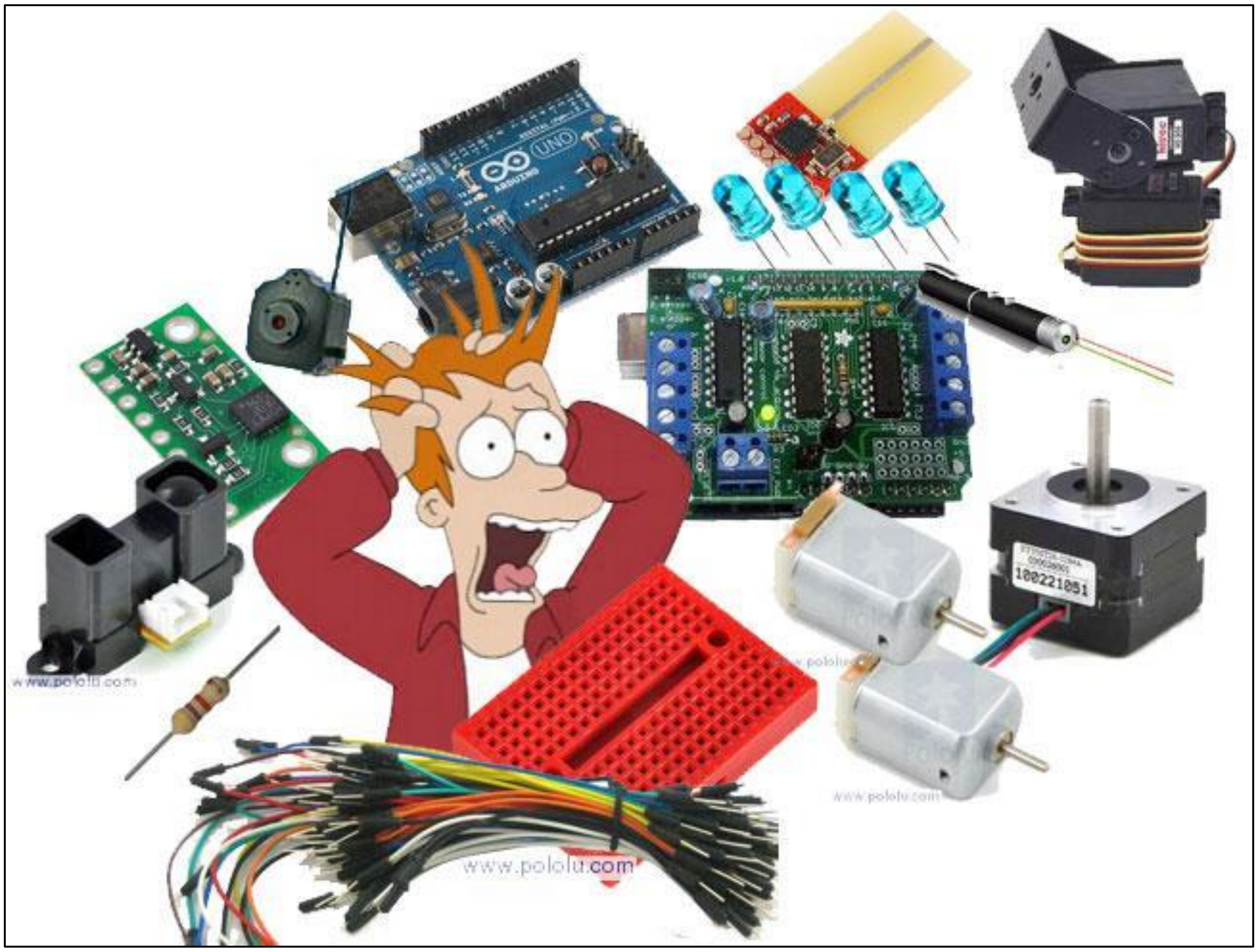


Table of Contents

1 Interfacing Sensors and Actuators
- 1.1 Reading
- 1.2 Introduction

2 Sensors
- 2.2 Digital Interface
  - 2.2.1 Design Example: DIP Switch
  - 2.2.2 Parallel Interface
- 2.3 Analog Interface
  - 2.3.1 Design Example

2.3.2 Voltage Range and Electromagnetic Interference (EMI)
3 Actuators
3.1 Digital Interface
3.3 Motor and Pulse Width Modulation (PWM) Interface
3.3.1 Motor On/Off
3.3.2 Change Motor Direction
3.3.3 Change the Speed of a Motor
3.3.4 All The Above
3.3.5 Servos
4 Serial Interface
4.1 Universal Asynchronous Receiver/Transmitter (USART)
4.2 Serial Peripheral Interface (SPI)
4.3 Inter-Integrated Circuit (I2C)
5 Recovering I/O Resources
6 Rover Example
6.1 Sensors
6.2 Actuators
7 Rover with Arduino Nano Example
7.1 Scan / Tilt Platform
7.1.1 Sensors
7.1.2 Actuators
7.1.3 Serial Communication
7.2 Rover
7.2.1 Sensors
7.2.2 Actuators
7.2.3 Serial Communication

1 Interfacing Sensors and Actuators

1.1 Reading

- Microcontroller Interfacing Circuits by Revolution Education Ltd.
- For help with a specific interface, or just to look for ideas, visit the related Arduino Forum.
- A detailed discussion of Pulse Width Modulation is beyond the scope of the document. We will be covering PWM in more detail later in the semester. For now you may want to read this nicely illustrated article: Working with Atmel AVR Microcontroller Basic Pulse Width Modulation (PWM) Peripheral

1.2 Introduction

In Part 1 we developed a Resource Map for our rover (Table 1.0). In this section we look at how sensors and actuators could be incorporated into the design of our rover. Specifically, we will map our sensors and actuators to our I/O pins.

Table 1.0 Rover Resource Summary

Up To	I/O Pin Names	Sensor/Actuator Type
6	Analog (ADC0 to ADC5)	Analog Sensors
10	GPIO (PC5-0, PB5, PB2, PB1, PD2)	Digital (on/off) Sensors
10	GPIO (PC5-0, PB5, PB2, PB1, PD2)	Digital (on/off) Actuators
2	PWM (PWM1A, PWM1B)	PWM Controlled Motors
4	H-Bridge Channels (M1, M2, M3, M4)	Bi-Directional PWM Controlled Motors
1	Serial Interface (SDA, SCL)	I^2C Sensors and/or Actuators

While **Part 1** of this System Interface Design discussion was from the perspective of the **ATmega328P**; **Part 2** will be presented from the perspective of the **I/O device**.

A detailed discussion of the ATmega328P subsystems, along with the software required to run them, is outside the scope of this document and will be covered in future lectures (hopefully).

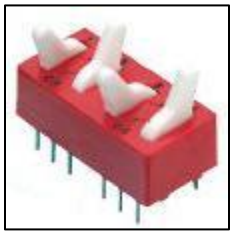
2 Sensors

2.2 Digital Interface

Sensors implementing a digital interface may be as simple as a micro-switch (on/off), DIP switch, or as complex as a digital camera. Consequently, you always need to **start with the data sheet** for your specific sensor(s).

2.2.1 Design Example: DIP Switch

A DIP switch is an example of a digital sensor which we could add to our design to allow our rover to determine the **mission phase** upon reset. Specifically, in the initialization section of the code the switch would be read and if zero the rover would begin mapping its surroundings and if one the rover would begin executing the mission.



Working from the schematic of the device, we wire the four (4) Single Pole Double Throw (SPDT) DIP switch(s) to the **GPIO Pins** (PC5-0, PB5, PB2, PB1, PD2) of the ATmega328P.

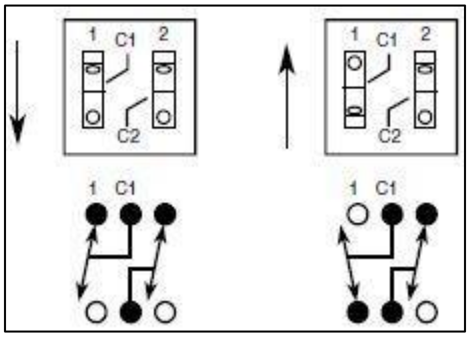


Figure 2.0 Two SPDT DIP Switches

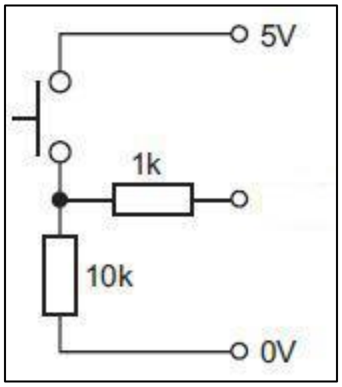


Figure 2.1 SPST Switch Schematic

2.2.2 Parallel Interface

For our rover we are limited to no more than 10 digital inputs (see Table 1.0). Consequently, while wiring 2 DIP switches to the ATmega328P is not a problem, while wiring a parallel device with a large word size is not. For example an 8-bit A/D converter would consume 80% of our available I/O resources. A much better solution would be an A/D converter that supports the I2C interface. Here we could get a 12-bit A/D converter with no loss in pin count.

2.3 Analog Interface

Many sensors output an analog voltage, including our long and medium range IR sensors. The ATmega328P has a single ADC subsystem whose input can come from up to 6 multiplexed channels (ADC0 to ADC5). The reference design preserves all 6 of these analog channels. From a practical standpoint, the term **multiplexed** means that although our design can support up to 6 analog sensors, we can only read one at a time.

2.3.1 Design Example

As with all devices, start with a schematic.

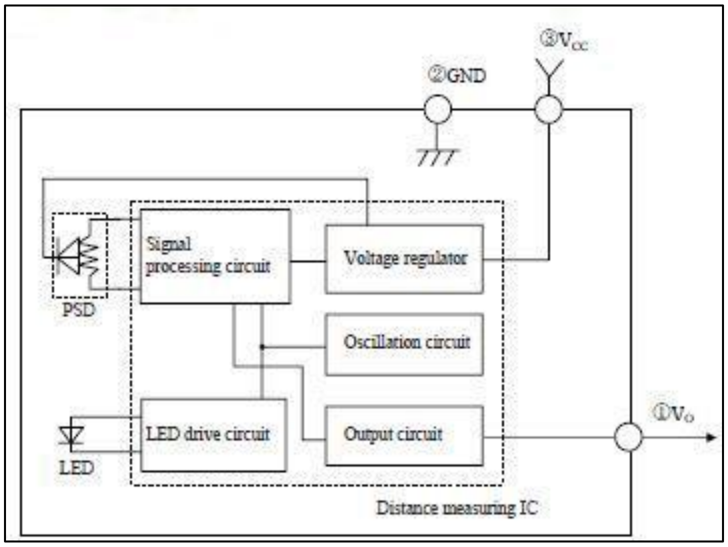


Figure 2.3 Medium Range IR Sensor Block Diagram

From the block diagram it is seen that the interface of an IR sensor is a single analog wire. In the absence of any other resource requirements, we are free to **wire** this sensor output **to any one of our six analog inputs** (ADC0 to ADC5).

2.3.2 Voltage Range and Electromagnetic Interference (EMI)

When working with any analog interface you should be sensitive to the output swing of the analog signal and to the introduction of noise.

For our medium range IR sensor, the **Voltage Output** peaks at around 3.1 V. That means to maximize our resolution we will want to use an external voltage reference of 3.3V. The full scale reading of the ADC can be set to the AVCC (5 v), AREF, or an internal 1.1v reference voltage. So in this case we will want to wire a 3.3v reference source to **AREF**. In a future lecture you will also find we need to place a **limiting resistor** between the our reference source and AREF.

Noise is almost always a concern when working with an analog signal. For our IR sensor the data sheet recommends a **10 uF capacitor** be placed as closely as possible to the sensor. In addition, analog signals are often sent over a twisted and shielded cable.

3 Actuators

3.1 Digital Interface

All ten available General Purpose I/O (GPIO) pins may be configured as outputs. The output circuit of the ATmega328 can sink or source a respectable **20 mA**. This means that the ATmega328P can directly turn on/off LED indicators, without the need for an external driver (you will still need a limiting resistor).

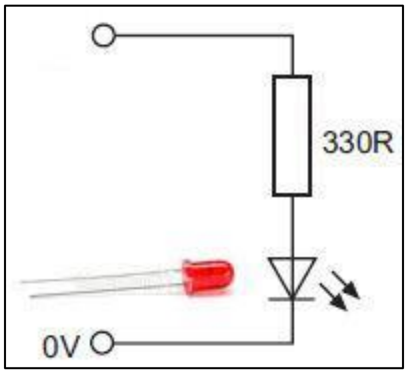


Figure 3.0 Diode Circuit

Another, digital actuator is our 650nm 1mW 8x13mm Laser Module. Once again before you purchase a device, make sure you have a datasheet. In the case of an laser, you need to know if it includes a current source, in which case you only need to turn the laser on/off, Otherwise, you will need to design in your own current source..

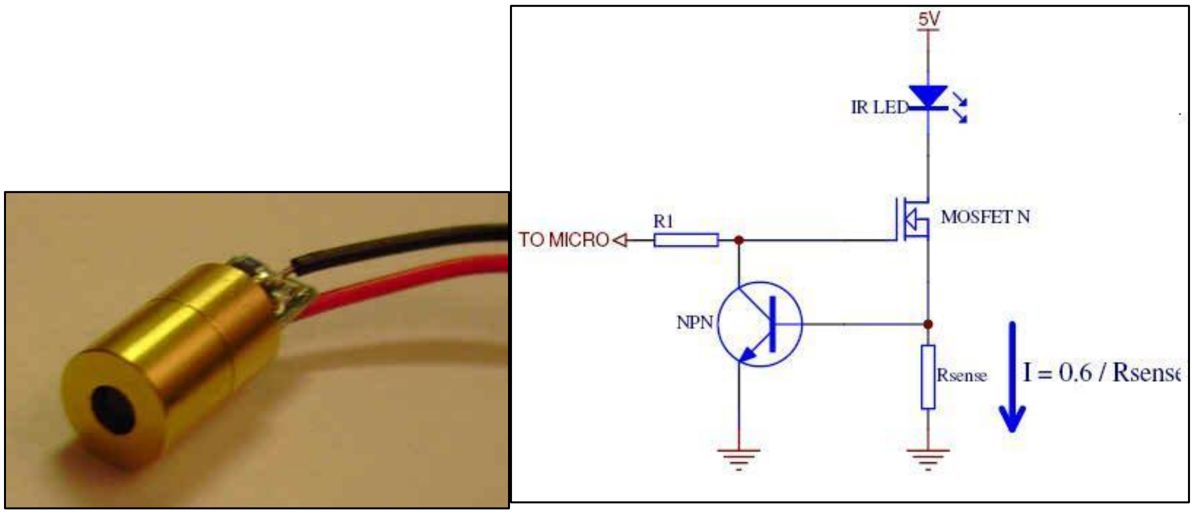


Figure 3.1 Laser Diode Constant Current Circuit
source: Sparkfun Forum

3.3 Motor and Pulse Width Modulation (PWM) Interface

3.3.1 Motor On/Off

If you only need to turn a DC motor ON or OFF you can use any one of the ten GPIO pins. In any case you will need to add a drive circuit. This can be as simple as a single MOSFET transistor, pull-down resistor, and a flyback diode.

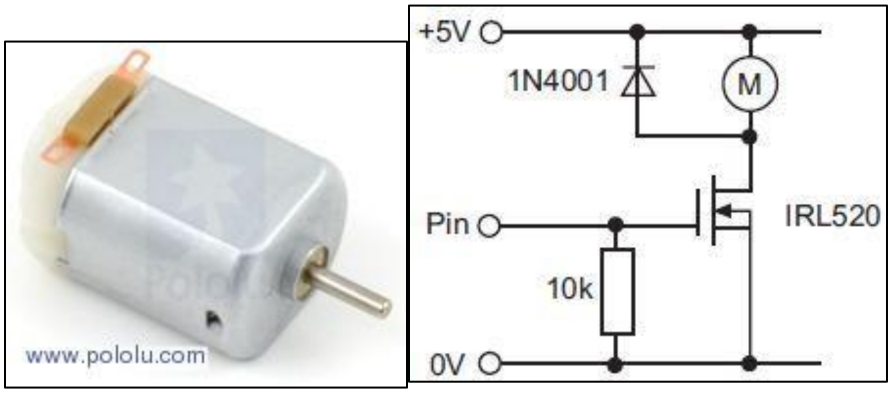


Figure 3.2 MOSFET Drive Circuit

3.3.2 Change Motor Direction

If you want to change the direction that the motor rotates, you will need to add an H-Bridge. Each H-bridge requires at least **two** (2) **GPIO pins** (Rotate Clockwise, Rotate Counterclockwise, Off).

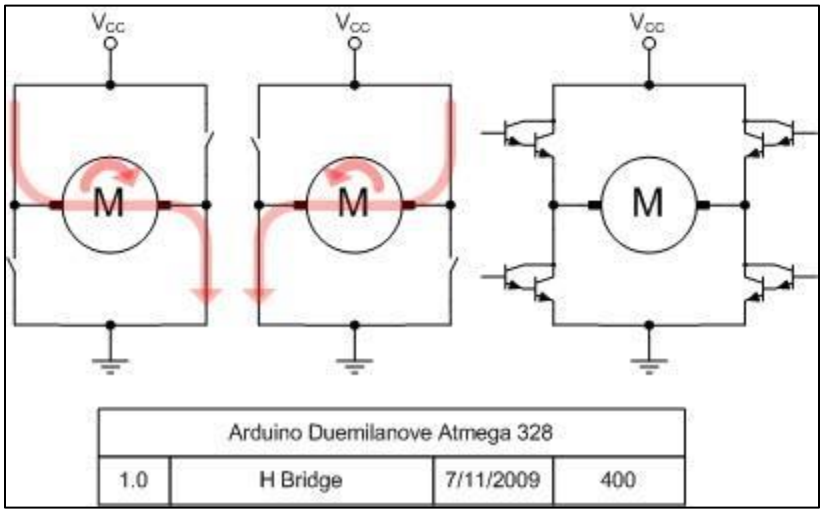


Figure 3.3 H-Bridge

3.3.3 Change the Speed of a Motor

A simple variable resistor is all you need if you want to control the speed of a DC motor manually. To control the speed of a DC motor with a microcontroller you will use one of our six PWM channels (e.g., PWM1A, PWM1B).

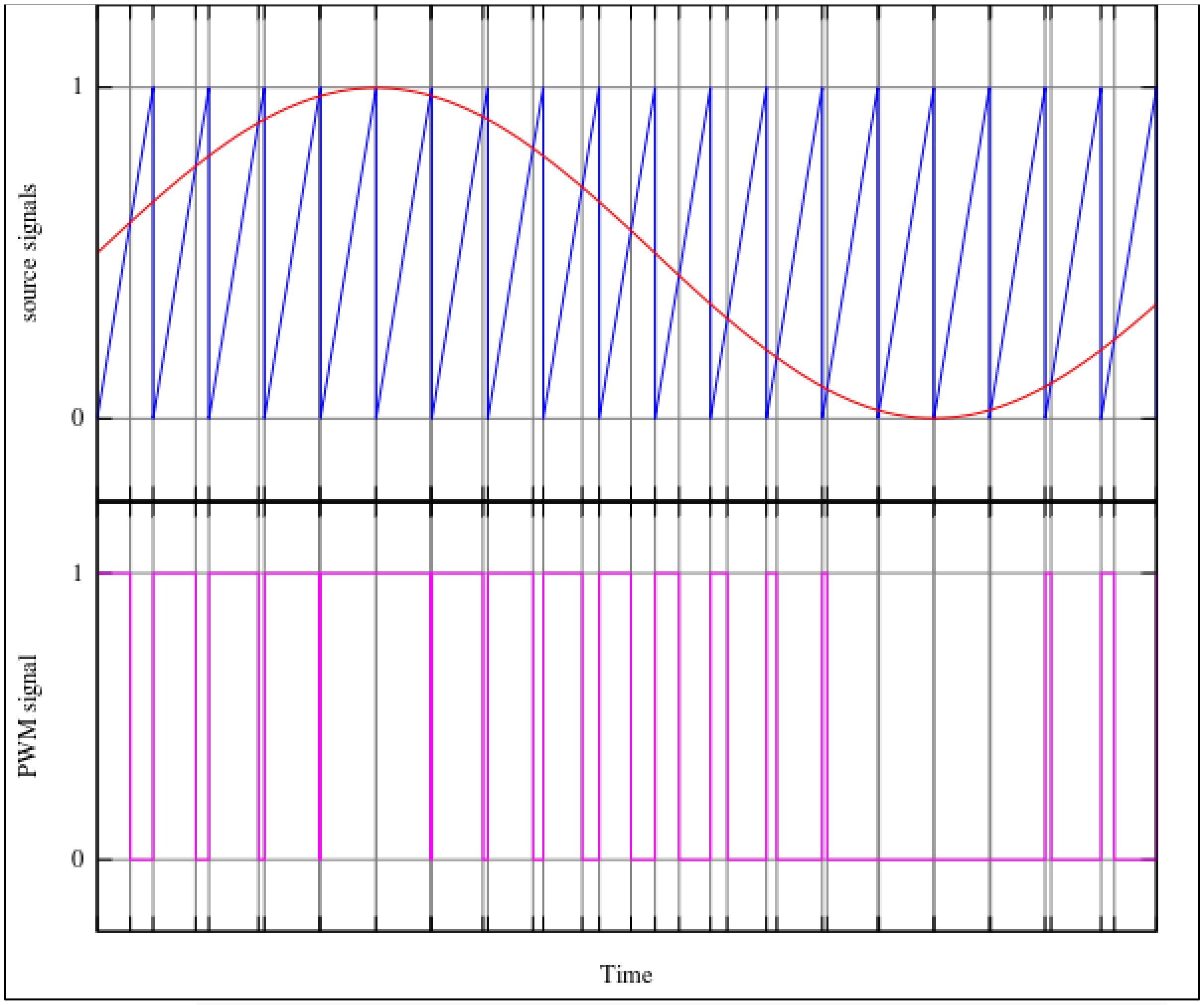
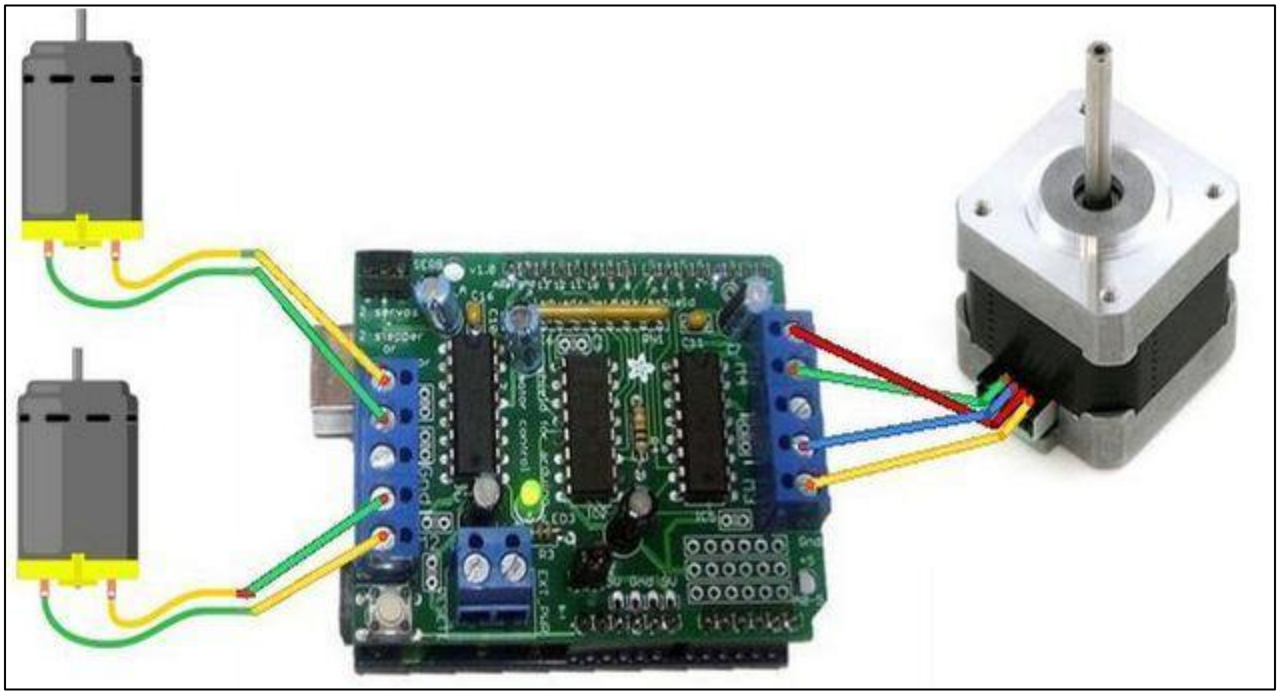


Figure 3.4 A simple method to generate the PWM pulse train corresponding to a given signal is the intersective PWM: the signal (here the red sinewave) is compared with a sawtooth waveform (blue). When the latter is less than the former, the PWM signal (magenta) is in high state (1). Otherwise it is in the low state (0).: Wikipedia

3.3.4 All The Above

If you want to turn your DC motor on/off, change direction, and control the speed of the motor you will need two (2) GPIO pins to configure the H-Bridge and a single PWM channel to control the speed. Without the Adafruit motor shield, If you wanted to control 4 DC motors (or 2 DC motors and a Bi-polar stepper motor) you would need eight (8) GPIO pins plus 4 PWM channels. With the motor shield we only lost 4 GPIO pins - we still need 4 PWM channels.

Attaching a DC motor to the Adafruit motor shield is as simple as wiring your DC motor to one of the four (4) Motor control connector pairs.



3.3.5 Servos

Reprinted from : Microcontroller Interfacing Circuits by Revolution Education Ltd.

A typical servo has just three connection wires, normally red, black and white (or yellow). The red wire is the 5V supply, the black wire is the 0V supply, and the white (or yellow) wire is for the positioning signal. The positioning signal is a pulse between **0.75 and 2.25 milliseconds** (ms) long, repeated about every **18 ms** (so there are roughly 50 pulses per second). With a 0.75ms pulse the servo moves to one end of its range, and with a 2.25ms pulse the servo moves to the other. Therefore, with a **1.5 ms pulse**, the servo will move to the central position. If the pulses are stopped the servo will move freely to any position. Unfortunately servos require a large current (up to 1A) and also introduce a large amount of noise on the power rail. **Therefore as with all motors, the servo should be powered from a separate power supply.** Remember that when using two power supplies the two ground rails must be joined to provide a common reference point.

4 Serial Interface

The ATmega328P supports three serial interface protocols: Universal Asynchronous Receiver/Transmitter (USART), Serial Peripheral Interface (SPI), and Inter-Integrated Circuit (I2C). All three support two-way communications. This means that all three serial subsystems of the ATmega328P can work as easily with actuators as sensors (see Section 2.1) which implement one or more of these interface types.

4.1 Universal Asynchronous Receiver/Transmitter (USART)

For the purposes of this study, the USART will be reserved for communications between the rover and the PC. As we learned in Part 1, the USART subsystem uses two IC pins (TXD and RXD).

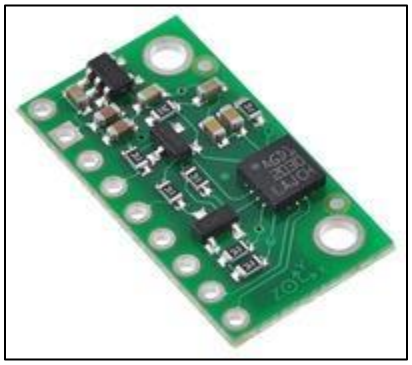
4.2 Serial Peripheral Interface (SPI)

The SPI Subsystem of the ATmega328P requires four pins to implement 2-way serial communications (**SCK, MISO, MOSI, SS**). These lines are are wired to pins PB5 (**SCK**/PCINT5), PB4 (**MISO**/PCINT4), PB3 (**MOSI**/**OC2A**/PCINT3), and PB2 (**SS**/**OC1B**/PCINT2). Of interest here are Output Compare signals **OC2A** and **OC1B**. These signals are from the 8-bit Timer 2 and 16-bit Timer 1 subsystems. Both timer subsystems are used by the Adafruit motor shield to generate Pulse Width Modulated signals PWM2A and PWM1B. Pulse Width Modulation is critical for controlling the speed of DC motors (PWM2A) and setting the angle of a servo (PWM1B). Consequently, Adafruit implemented their SPI interface with General Purpose I/O (GPIO) ports and software. In this way they were able to maximize the number of PWM signals available to the shield, while sacrificing the SPI subsystem of the ATmega328P.

So what if you are working with a peripheral device that implements the SPI serial communications protocol? First, you can follow the Adafruit path and implement your SPI interface in software. Second, the SPI subsystem can be recovered at the cost of some functionality or the time sharing of shared resources. An example of the last case would be to communicate with your SPI device only when the shared motor control signal was not required (motor is off).

4.3 Inter-Integrated Circuit (I2C)

The I2C or TWI in Atmel speak, is a serial communications protocol with similar functionality to the SPI communications protocol. However, unlike the SPI which requires 4 pins to implement two-way communications, the I2C needs only two pins (**SDA, SCL**). The trade-off here is in the added complexity of the I2C interface. Today many intelligent sensors and actuators support both the I2C and SPI interface. This is the case with our L3G4200D 3-Axis Gyro Carrier with Voltage Regulator. As illustrated in Table 1.0 , the pins of the ATmega328P I2C subsystem are available for our sensors and actuators.



It should be noted that the I2C interface supports up to 128 devices without the need for any additional I/O pins.

5 Recovering I/O Resources

TBS

6 Rover Example

For our Rover we have the following Sensors and Actuators.

6.1 Sensors

I2C serial interface communicating with 3-Axis Gyroscope
I2C serial interface communicating with Current Sensor (optional)
I2C Arduino Nano (optional)
Analog input from Mid Range IR
Analog input from Long Range IR
Analog input from RC Circuit to 7.2V NiCD Battery – Dirty Power
Analog input from RC Circuit to 9V NiMH Battery – Clean Digital
Analog input from RC Circuit to 9V NiMH Battery – Camera
Four (4) Digital inputs from two (2) Shaft Encoders. *Assumes full resolution of quadrature shaft encoders*.

6.2 Actuators

Two (2) PWM outputs to H-Bridges controlling bipolar Stepper Motor – Adafruit Motorshield
Two (2) PWM outputs to H-Bridge controlling two DC Motors – Adafruit Motorshield
One PWM output to Servo – Adafruit Motorshield
One Digital output thru a Transistor circuit to turn the Laser on/off
One Digital output thru a Transistor circuit to turn the Camera on/off
(One or more Digital outputs to address bits of a 4051 Analog MUX for camera battery)

For our example we begin by allocating resources interfaced to ATmega328P peripheral subsystems requiring the use of only one set of pins. In this case the I2C serial interface. Looking at the matrix we see that this first step removes two Analog inputs leaving us with four (4). We need five (5) Analog Inputs so we are already in trouble. We will leave the sensors and actuators wired to the GPIO peripheral subsystem to the end. This is because these are the simplest to assign.The PWM channels are output from the Adafruit Motor Shield and so easily assigned. Once again we have very few options with respect to which pins can be used. We are now left with 4 Digital Input and 2 Digital Output pins to be connected to the GPIO peripheral subsystem of the ATmega328P. So we need to find 6 (4 inputs + 2 outputs) GPIO pins. Looking at the matrix we see that we have 3 GPIO pins left. So we are three (3) I/O pins short. How we find the 1 analog and 3 GPIO pins is left up to you. Hint: Look-up 4051, 74HCT595, and parallel shift registers like the 74HC166, 74HC194, and CD4014.

Table 2.0 System Resource Map

	ATmega328P	Arduino	Motor Shield	Motor Shield DIR (SPI Interface)	Motor Shield PWM (H-Bridge)	Rover
1	PD0 (RXD)	J1-1 Digital Pin 0				
2	PD1 (TXD)	J1-2 Digital Pin 1				
3	PD2 (INT0)	J1-3 Digital Pin 2	JP3 Pin 1			1.
4	PD3 (INT1, OC2B)	J1-4 Digital Pin 3	PWM2B		IC1 Pin 9 (3-4EN)	
5	PD4 (T0)	J1-5 Digital Pin 4	DIR_CLK	IC3 Pin 11 (SCK)		
6	PD5 (T1, OC0B)	J1-6 Digital Pin 5	PWM0B		IC2 Pin 1 (1-2)EN	
7	PD6 (OC0A, AIN0)	J1-7 Digital Pin 6	PWM0A		IC2 Pin 9 (3-4EN)	
8	PD7 (AIN1)	J1-8 Digital Pin 7	DIR_EN	IC3 Pin 13 (G)		
9	PC0 (ADC0)	J2-1 Analog Pin 0	JP5-1			Long Range IR
10	PC1 (ADC1)	J2-2 Analog Pin 1	JP5-2			Meduim Range IR
11	PC2 (ADC2)	J2-3 Analog Pin 2	JP5-3			9v NiMH Battery
12	PC3 (ADC 3)	J2-4 Analog Pin 3	JP5-4			7.2v NiCD Battery
13	PC4 (ADC 4, SDA)	J2-5 Analog Pin 4	JP5-5			I2C Gyro, Current Sensor
14	PC5 (ADC 5, SCL)	J2-6 Analog Pin 5	JP5-6			I2C Gyro, Current Sensor
15	PB0 (ICP1)	J3-1 Digital Pin 8	DIR_SER	IC3 Pin 14 (SER)		
16	PB1 (OC1A)	J3-2 Digital Pin 9	PWM1A		Servo 2	2.
17	PB2 (SS, OC1A)	J3-3 Digital Pin 10	PWM1B		Servo 1	Servo

18	PB3 (MOSI, OC2A)	J3-4 Digital Pin 11	PWM2A		IC1 Pin 1 (1-2EN)	
19	PB4 (MISO)	J3-5 Digital Pin 12	DIR _Latch	IC3 Pin 12 (RCK)		
20	PB5 (SCK)	J3-6 Digital Pin 13				3.
21	GND	J3-7 GND	GND			
22	AREF	J3-8 AREF				3.3 v through Resistor
23			X1-P1 (M1)			DC Motor Left
24			X1-P2 (M1)			DC Motor Left
25			X1-P3 (GND)			
26			X1-P4 (M2)			DC Motor Right
27			X1-P5 (M2)			DC Motor Right
28			X2-P1 (M4)			Stepper Motor Red
29			X2-P2 (M4)			Stepper Motor Blue
30			X2-P3 (GND)			
31			X2-P4 (M3)			Stepper Motor Black
32			X2-P5 (M3)			Stepper Motor Green

7 Rover with Arduino Nano Example

In this design example to the chassis mounted Arduino Uno we add a Arduino Nano to the scan/tilt platform.

7.1 Scan / Tilt Platform

7.1.1 Sensors

Analog input from Mid Range IR
Analog input from Long Range IR
Analog input from RC Circuit to 9V NiMH Battery – Camera

7.1.2 Actuators

One PWM output to Servo – Adafruit Motorshield
One Digital output thru a Transistor circuit to turn the Laser on/off
One Digital output thru a Transistor circuit to turn the Camera on/off

7.1.3 Serial Communication

I2C Interface to Rover MCU

7.2 Rover

7.2.1 Sensors

Analog input from RC Circuit to 7.2V NiCD Battery – Dirty Power
Analog input from RC Circuit to 9V NiMH Battery – Clean Digital
Four (4) Digital inputs from two (2) Shaft Encoders.

7.2.2 Actuators

Two (2) PWM outputs to H-Bridges controlling bipolar Stepper Motor – Adafruit Motorshield
Two (2) PWM outputs to H-Bridge controlling two DC Motors – Adafruit Motorshield

7.2.3 Serial Communication

I2C Interface to Scan/Tilt MCU
I2C serial interface communicating with 3-Axis Gyroscope
I2C serial interface communicating with Current Sensor

For our example we begin by allocating resources interfaced to ATmega328P peripheral subsystems requiring the use of only one set of pins. In this case the I2C serial interface. Looking at the matrix we see that this first step removes two Analog inputs leaving us with four (4). We need two (2) Analog Inputs so we have two (2) pins which may be used for sensors and actuators to be wired to the GPIO peripheral subsystem (PC 2 - 5). Four (4) of our six (6) PWM channels are required by the Adafruit Motor Shield. This leaves us with two (2) more pins which may be used for sensors and actuators to be wired to the GPIO peripheral subsystem. Looking at the matrix we see that we have 2 GPIO pins PD2 and PB5 which are unused, giving us a total of six (6) pins available to the GPIO peripheral subsystem. We only need four (4) for our two shaft encoders which means we have two (2) spare pins.

Table 3.0 System Resource Map for Chassis Mounted Arduino Uno

	ATmega328P	Arduino	Motor Shield	Motor Shield DIR (SPI Interface)	Motor Shield PWM (H-Bridge)	Rover
1	PD0 (RXD)	J1-1 Digital Pin 0				
2	PD1 (TXD)	J1-2 Digital Pin 1				
3	PD2 (INT0)	J1-3 Digital Pin 2	JP3 Pin 1			1.
4	PD3 (INT1, OC2B)	J1-4 Digital Pin 3	PWM2B		IC1 Pin 9 (3-4EN)	
5	PD4 (T0)	J1-5 Digital Pin 4	DIR_CLK	IC3 Pin 11 (SCK)		
6	PD5 (T1, OC0B)	J1-6 Digital Pin 5	PWM0B		IC2 Pin 1 (1-2)EN	
7	PD6 (OC0A, AIN0)	J1-7 Digital Pin 6	PWM0A		IC2 Pin 9 (3-4EN)	
8	PD7 (AIN1)	J1-8 Digital Pin 7	DIR_EN	IC3 Pin 13 (G)		
9	PC0 (ADC0)	J2-1 Analog Pin 0	JP5-1			2.
10	PC1 (ADC1)	J2-2 Analog Pin 1	JP5-2			3.
11	PC2 (ADC2)	J2-3 Analog Pin 2	JP5-3			9v NiMH Battery
12	PC3 (ADC 3)	J2-4 Analog Pin 3	JP5-4			7.2v NiCD Battery
13	PC4 (ADC 4, SDA)	J2-5 Analog Pin 4	JP5-5			I2C Gyro, Current Sensor, Scan/Tilt
14	PC5 (ADC 5, SCL)	J2-6 Analog Pin 5	JP5-6			I2C Gyro, Current Sensor, Scan/Tilt
15	PB0 (ICP1)	J3-1 Digital Pin 8	DIR_SER	IC3 Pin 14 (SER)		
16	PB1 (OC1A)	J3-2 Digital Pin 9	PWM1A		Servo 2	4.
17	PB2 (SS, OC1A)	J3-3 Digital Pin 10	PWM1B		Servo 1	5.

18	PB3 (MOSI, OC2A)	J3-4 Digital Pin 11	PWM2A		IC1 Pin 1 (1-2EN)	
19	PB4 (MISO)	J3-5 Digital Pin 12	DIR _Latch	IC3 Pin 12 (RCK)		
20	PB5 (SCK)	J3-6 Digital Pin 13				6.
21	GND	J3-7 GND	GND			
22	AREF	J3-8 AREF				
23			X1-P1 (M1)			DC Motor Left
24			X1-P2 (M1)			DC Motor Left
25			X1-P3 (GND)			
26			X1-P4 (M2)			DC Motor Right
27			X1-P5 (M2)			DC Motor Right
28			X2-P1 (M4)			Stepper Motor Red
29			X2-P2 (M4)			Stepper Motor Blue
30			X2-P3 (GND)			
31			X2-P4 (M3)			Stepper Motor Black
32			X2-P5 (M3)			Stepper Motor Green