Notes

• Starting Experiment 4 this week – “Robot Speed and Position Control”
• This lab requires you to have BOTH wheels running by the Lab 4 demo date
• Tuesday this week
  – Turn in prelab 4.A and complete 4.A
  – Redo Labs 2B and 3 on the second wheel
• Thursday this week:
  – Have speed control working on both wheels
  – Start 4.B: connecting Arduino microcontroller board to your speed control circuitry and coding the microcontroller
• Lab 4 demo is Thursday the week after (see Lab calendar)
Experiment 4

Experiment 3a

- 0-10V
- Rb1
- 10VDC
- Q3
- Q1
- Q4
- Q2
- GND
- VDC+
- VDC-

Forward Controller
Reverse Controller

NOT Forward
Forward

Speed error

0-5V Speed Output proportional to actual motor speed

Experiment 3b

0-5V Speed Output proportional to desired motor speed

Experiment 1 & 2

Speed Sensor and Filter

5VDC

GND

wheel

ENCA

V_{speed}

Electronics Design Laboratory
Three power inputs: USB (provides 5VDC), VIN pin (7-12VDC) and a barrel jack (9-12VDC)
• Need to select which power input to use, and generate 5VDC and 3.3VDC
Either batteries or a DC Power Supply will generate 10VDC, which we will connect to the VIN pin.

Make sure to use decoupling capacitors on the 10VDC line.

Not using decoupling caps will waste a bunch of your time since motors are noisy and may cause trouble for your microcontroller if the power supply is not clean.
The Arduino is built around an 8-bit microcontroller. Communication with the ATmega328P is achieved through a USB interface. Since the ATmega328P does not ‘talk’ USB, a serial to USB conversion must be made.
The Arduino is built around an 8-bit microcontroller. Communication with the ATmega328P is achieved through a USB interface. Since the ATmega328P does not ‘talk’ USB, a serial to USB conversion must be made.
Digital Inputs/Outputs: GPIO

GPIO: General Purpose Inputs/Outputs. These are digital outputs, so only voltages of 0V or 5V may be generated. Pins marked with (~) may be used to output PWM signals. Maximum current of 40mA per output. Pins default to Inputs

**Internal ATmega328P Hardware**

- **INPUT/OUTPUT**
- **PULLUP**
- GPOIX 0V-5V
- 20kΩ
- 100MΩ
- 5V
- {0V,5V}
- {0,1}

**Program Code**

- `digitalRead(GPIOX)`
- `digitalWrite(GPIOX, {HIGH, LOW})`
- `pinMode(GPIOX, mode)`
GPIO: General Purpose Inputs/Outputs. These are digital outputs, so only voltages of 0V or 5V may be generated. Pins marked with (~) may be used to output PWM signals. Maximum current of 40mA per output. Pins default to Inputs

As Digital Inputs...

As Digital Outputs...

As PWM Outputs...

analogWrite(GPIOX, duty_cycle) = \( \left( \frac{\text{duty_cycle}}{255} \right) \times T_{\text{PWM}} \)

\( \text{duty_cycle} = \text{Integer} \left( 255 \times \frac{t_{\text{on}}}{T_{\text{PWM}}} \right) \)
Analog Inputs

Analog Inputs: 6 Analog inputs are available. These pins convert an analog voltage in the range 0V-AREF to an integer in the range 0-1023. The value read by the ATmega328 can be calculated as:

\[
VALX = \text{Integer}\left[ \frac{V_A}{AREF} \cdot (1023) \right]
\]

Analog reference AREF defaults to 5V. It takes approximately 100 \( \mu \)s to preform an analog read. Analog inputs A0-A5 can be configured to act exactly like the GPIO pins if needed.

Internal ATmega328P Hardware

Program Code

\begin{itemize}
  \item analogReference(mode)
  \item analogRead(AX)
\end{itemize}
Analog Inputs: 6 Analog inputs are available. These pins convert an analog voltage in the range 0V-AREF to an integer in the range 0-1023. The value read by the ATmega328 can be calculated as:

$$\text{VALX} = \text{Integer} \left[ \frac{V_A}{\text{AREF}} \left(1023\right) \right]$$

Analog reference AREF defaults to 5V. It takes approximately 100 μs to perform an analog read. Analog inputs A0-A5 can be configured to act exactly like the GPIO pins if needed.

As Analog Inputs...

Sample 1: 2.6V
Sample 2: 2.7V
Sample 3: 1.3V

\(\text{analogRead}(AX)\) returns

\[\text{Integer}\left[\left(\frac{2.6V}{5.0V}\right)1023=\left(\frac{2.6V}{5.0V}\right)1023\right]=531\]

\(\text{analogRead}(AX)\) returns 552
\(\text{analogRead}(AX)\) returns 266
GPIO: General Purpose Inputs/Outputs. These are digital outputs, so only voltages of 0V or 5V may be generated. Pins marked with (~) may be used to output PWM signals. Maximum current of 40mA per output. Pins default to Inputs

USB input used for programming and debugging

Analog Inputs: 6 Analog inputs are available. These pins convert an analog voltage in the range 0V-AREF to an integer in the range 0-1023

$$\text{VALX} = \text{Integer}\left[ \frac{V_A}{\text{AREF}} \right] (1023)$$

Power system connected as seen previously
Robot Control with the Arduino

- Outputs to robot: **Stop/Go controls** (2 per wheel) **Speed reference** (1 or 2 total)
- Voltages: **5VDC** for speed sensing circuits and **10VDC** for Motors/Encoders
- Inputs from robot: **On/Off** and **Encoder Pulses**
Programming

• Plug-in USB cable
• First-time use only: install USB driver (if need be)
• Start Arduino development environment
• Open an example: File→Examples→1.Basics→Blink
• Click Upload
• http://arduino.cc
  Extensive on-line reference, examples, blog, forum, ...

```cpp
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.
*/

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

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
}```
“Blink” example

/**
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

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
}
Speed Control Setup for One Wheel

Pin 7
- Logic High (+5V) activates clockwise
- 1K

Pin 8
- Logic High (+5V) activates counter-clockwise
- 1K

Pin 9
- Speed Reference (0-5V 500 Hz PWM from waveform gen or from Arduino)
- +5V 1K

Pin 6
- ON/OFF switch
- 1K

Power supplies:
- +10V from bench power supply (or from battery)
- +5V from Arduino board
/*
ECEN2270 motor speed control example (left wheel only)
*/

// define pins
const int pinON = 6;  // connect pin 6 to ON/OFF switch, active HIGH
const int pinCW_Left = 7;  // connect pin 7 to clock-wise PMOS gate
const int pinCC_Left = 8;  // connect pin 8 to counter-clock-wise PMOS gate
const int pinSpeed_Left = 9;  // connect pin 9 to speed reference

// setup pins and initial values
void setup() {
    pinMode(pinON, INPUT);
    pinMode(pinCW_Left, OUTPUT);
    pinMode(pinCC_Left, OUTPUT);
    pinMode(pinSpeed_Left, OUTPUT);
    pinMode(13, OUTPUT);  // on-board LED
    digitalWrite(pinCW_Left, LOW);  // stop clockwise
    digitalWrite(pinCC_Left, LOW);  // stop counter-clockwise
    analogWrite(pinSpeed_Left, 100);  // set speed reference, duty-cycle = 100/255
}

void loop() {
    digitalWrite(13, LOW);  // turn LED off
    do {} while (digitalRead(pinON) == LOW);  // wait for ON switch
    digitalWrite(13, HIGH);  // turn LED on
    delay(1000);  // wait 1 second
    digitalWrite(pinCW_Left, HIGH);  // go clockwise
    delay(3000);  // for 3 seconds
    digitalWrite(pinCW_Left, LOW);  // stop
Lab 4 Part A

- **Clean up Lab 3 circuit**, make sure you have a speed sensor and two feedback controllers for **both wheels**
- **Play with the Arduino**. Test the Blink program, and run any other test programs that look interesting
- At the end of part A, you should be able to use the Arduino for speed control of both wheels
- Make sure you know which direction the wheels will spin!