Electricity/Electronics Review

• Current
• Voltage
• Resistance
• Ohm’s Law
• Power
<table>
<thead>
<tr>
<th><strong>Voltage</strong></th>
<th><strong>Current</strong></th>
<th><strong>Resistance</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>I</td>
<td>R</td>
</tr>
<tr>
<td>• The amount of potential across any two points in a circuit</td>
<td>• The rate of flow of charge past any point in a circuit.</td>
<td>• Opposition to charge flow.</td>
</tr>
<tr>
<td>• Volts (V)</td>
<td>• Amperes (A)</td>
<td>• Units: Ohms (Ω)</td>
</tr>
</tbody>
</table>
Water Analogy

High Current

Low Current
Big Pipe == Lower Resistance

\[ V = I R \]

Small Pipe == Higher Resistance

\[ V = I R \]
Measuring Voltage

Voltage is a measure of electrical potential energy. A voltage is also called a potential difference – it is measured across two points in a circuit – across a device.
Measuring Current

In order to measure this – you must break the circuit and insert the meter *in series* with the rest of the circuit.
Measuring Resistance

Components should be removed entirely from the circuit to measure resistance. Note the settings on the multimeter. Make sure that you are set for the appropriate range.
Analog Signals and Digital Signals
Prototyping Circuits using a Solderless Breadboard
Transistor

Diode

220 Ω or 330 Ω resistor

LWTL: DC Motor
Anatomy of an LED:

- Epoxy lens/case
- Wire bond
- Reflective cavity
- Semiconductor die
- Anvil Post
- Leadframe
- Flat

Symbol:

- Anode
- Cathode

Diagram:

- Anode
- Cathode
- p-type silicon
- n-type silicon
Semiconductors

- Semiconductors are materials that fall between conductors and insulators.
- They may act as insulators in some conditions and as conductors in others.
- Semiconductors can be doped; this is when another substance is added to the semiconductor to change its properties.
- Donor dopants produce an excess of electrons in the semiconductor. Semiconductors doped with donors are called n-type.
- Acceptor dopants produce an excess of positive “holes” where there are no electrons. Semiconductors doped with acceptors are called p-type.
Diodes

• A diode is a circuit element which essentially is a resistor with polarity; it has a different resistance in one direction than in the other.

• Most diodes have no resistance in one direction and very high resistance in the other, so that they only allow current to flow in one direction. These diodes are called rectifiers.

• Recall that semiconductors may change from insulators to conductors under certain conditions. For semiconductor diodes, the diode behaves as an insulator until a certain voltage is achieved across the diode. It then behaves as a conductor, allowing current to pass. When this happens, the diode is forward-biased.

• The symbol for a diode looks like an arrow that points in the direction of current flow. The diode shown below would allow current to flow from left to right.
Transistors

- Transistors are circuit components made of semiconductors that amplify and switch currents.
- A good example of how transistors work is the Bipolar Junction Transistor (BJT). In the NPN BJT, a layer of p-type semiconductor separates two sections of n-type semiconductor. When there is a voltage across the two n-type layers, no current can pass through. When positive voltage is applied to the p-type layer, however, the transistor becomes conductive, and current can pass through.
- In PNP transistors, two p-type semiconductors are separated by n-type semiconductor material. When positive voltage is applied to the n-type layer, it is closed; when negative voltage is applied, it is open.
Parts of a Transistor

- The terminal that receives current is called the collector.
- The terminal that releases current is called the emitter.
- The terminal that controls whether the transistor is on is called the base.
Integrated Circuits.
An integrated circuit (also referred to as an IC, a chip, or a microchip) is a system of electronic circuits on a small flat ‘chip’ of semiconductor material, usually silicon.

The integration of large numbers of transistors onto a single chip results in systems that are much smaller, faster and less costly than those made of discrete components.
A single-board microcontroller is a controller built onto a single printed circuit board. It provides all of the circuitry necessary for a useful control task:

- a microprocessor,
- I/O circuits,
- a clock generator,
- RAM,
- and any necessary support ICs

A Single-chip microcontroller integrates many of the features of single-board microcontrollers onto a single chip.
Analog INPUTS

Digital I/O
PWM(3, 5, 6, 9, 10, 11)

PWR IN
USB (to Computer)

RESET

SCL\SDA (I2C Bus)

POWER
5V / 3.3V / GND

Analog INPUTS

Digital I/O
PWM(3, 5, 6, 9, 10, 11)
<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller</td>
<td>ATmega2560</td>
</tr>
<tr>
<td>Operating Voltage</td>
<td>5V</td>
</tr>
<tr>
<td>Input Voltage (recommended)</td>
<td>7-12V</td>
</tr>
<tr>
<td>Input Voltage (limits)</td>
<td>6-20V</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>54 (14 for PWM output)</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>16</td>
</tr>
<tr>
<td>DC Current per I/O Pin</td>
<td>40 mA</td>
</tr>
<tr>
<td>DC Current for 3.3V Pin</td>
<td>50 mA</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>256KB/8KB for bootloader</td>
</tr>
<tr>
<td>SRAM</td>
<td>8 KB</td>
</tr>
<tr>
<td>EEPROM</td>
<td>4 KB</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>16 MHz</td>
</tr>
</tbody>
</table>
Arduino
Integrated Development Environment (IDE)

Two required functions / methods / routines:

```c
void setup()
{
    // runs once
}

void loop()
{
    // repeats
}
```
Your computer communicates to the Arduino microcontroller via a serial port through a USB-Serial adapter.

Check to make sure that the drivers are properly installed.
Next, double-check that the proper board is selected under the Tools → Board menu.
Three commands to know...

```c
pinMode(pin, INPUT/OUTPUT);
   ex: pinMode(13, OUTPUT);

digitalWrite(pin, HIGH/LOW);
   ex: digitalWrite(13, HIGH);

delay(time_ms);
   ex: delay(2500); // delay of 2.5 sec.
```

// NOTE: -> commands are CASE-sensitive
int sensorValue;
int ledPin;

void setup()
{
    // put your setup code here, to run once:
    int setupVariable;
}

void loop()
{
    // put your main code here, to run repeatedly:
    int loopScopeVariable
}
// Projet 2 - Traffic Lights

int redLedDelay = 10000;
int greenLedDelay = 10000;
int redLed = 7;
int yellowLed = 6;
int greenLed = 5;

void setup(){
    pinMode(redLed, OUTPUT);
    pinMode(yellowLed, OUTPUT);
    pinMode(greenLed, OUTPUT);
}

void loop(){

    digitalWrite(redLed, HIGH);
    delay(redLedDelay);

    digitalWrite(yellowLed, HIGH);
    delay(2000);

    digitalWrite(redLed, LOW);
    digitalWrite(yellowLed, LOW);
    digitalWrite(greenLed, HIGH);
    delay(greenLedDelay);

    digitalWrite(yellowLed, HIGH);
    digitalWrite(greenLed, LOW);
    delay(2000);

    digitalWrite(yellowLed, LOW);
}
Analog Output

- Can a digital device produce analog output?

- Analog output can be simulated using pulse width modulation (PWM)

Image from *Theory and Practice of Tangible User Interfaces* at UC Berkley
PWM Duty Cycle

output voltage = \((\text{on\_time} / \text{cycle\_time}) * 5V\)

Fixed cycle length; constant number of cycles/sec
PMW Pins

Your Arduino board has built in PWM circuits, on pins 3, 5, 6, 9, 10, and 11

- Command: `analogWrite(pin, value)`
- value is duty cycle: between 0 and 255
- Examples:
  - `analogWrite(9, 128)` for a 50% duty cycle
  - `analogWrite(11, 64)` for a 25% duty cycle
analogWrite(pin, val);

**pin** – refers to the OUTPUT pin (limited to pins 3, 5, 6, 9, 10, 11.) – denoted by a ~ symbol

**val** – 8 bit value (0 – 255).

0 => 0V | 255 => 5V
void setup() {
    // declare pin 9 to be an output:
    pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
    // set the brightness of pin 9:
    analogWrite(led, brightness);

    // change the brightness for next time through the loop:
    brightness = brightness + fadeAmount;

    // reverse the direction of the fading at the ends of the fade:
    if (brightness == 0 || brightness == 255) {
        fadeAmount = -fadeAmount;
    }

    // wait for 30 milliseconds to see the dimming effect
    delay(30);
}
PWM: Pulse Width Modulation

PWM simulates DC voltage control for slow loads

The effective voltage is

\[ V_{\text{eff}} = V_s \frac{\tau_o}{\tau_c} \]

\[ \frac{\tau_o}{\tau_c} \]

is called the duty cycle
Arduino PWM commands

Configure the output pin:

```cpp
PWM_pin = ...;  // one of 3, 5, 6, 9, 10, 11

void setup() {
    pinMode( PWM_pin, OUTPUT);
}
```

Set the duty cycle

```cpp
void loop() {
    int duty_cycle = 150;  // between 0 and 255

    analogWrite( PWM_pin, duty_cycle );
}
```

The duty cycle is an 8 bit value:

```
0 ≤ duty_cycle ≤ 255
```
Transistor as the switching device

• Each Arduino output channel has a 40 mA limit
• The maximum current draw for an Arduino is 200 mA
• Use Arduino as the brain
• Let another switching element be the brawn
Use an NPN Transistor as a switch

This device is designed for use as a medium power amplifier and switch requiring collector currents up to 500 mA.
A transistor allows on/off control to be automated and it allows switching of more current than an Arduino digital pin can supply.

Pin 9 or another PWM pin drives the transistor base.
Diode and transistor orientation

Orient the diode so that the silver stripe is at the same voltage as the positive motor terminal.

Collector: Connect to +5V
Base: Connect to motor control pin on Arduino
Emitter: Connect to positive terminal of motor
PWM signal is connected to transistor base
Arduino program to spin the DC Motor

Code is in spin_DC_motor.ino

```cpp
// spin_DC_motor.ino  Use PWM to control DC motor speed

int motorPin = 3;  // Pin 3 has PWM, connected it to the DC motor

void setup() {
  pinMode(motorPin, OUTPUT);  // Set motor pin to output mode
}

void loop() {
  analogWrite(motorPin, 150);  // Motor at 150/255 of full speed
  delay(1000);
  analogWrite(motorPin, 250);  // Motor at 250/255 of full speed
  delay(1000);
}
```
Digital Input

- Connect digital input to your Arduino using Pins # 0 – 13 (Although pins # 0 & 1 are also used for programming)

- Digital Input needs a `pinMode` command:
  ```c
  pinMode (pinNumber, INPUT);
  Make sure to use ALL CAPS for `INPUT`
  ```

- To get a digital reading:
  ```c
  int buttonState = digitalRead (pinNumber);
  ```

- Digital Input values are only HIGH (On) or LOW (Off)
Digital Sensors

• Digital sensors are more straightforward than Analog

• No matter what the sensor there are only two settings: On and Off

• Signal is always either HIGH (On) or LOW (Off)

• Voltage signal for HIGH will be a little less than 5V on your Uno

• Voltage signal for LOW will be 0V on most systems
We set it equal to the function `digitalRead(pushButton)`.

We declare a variable as an integer.

```
int buttonState = digitalRead(pushButton);
```

The function `digitalRead()` will return the value 1 or 0, depending on whether the button is being pressed or not being pressed.

We name it `buttonState`.

The value 1 or 0 will be saved in the variable `buttonState`.

Recall that the `pushButton` variable stores the number 2.
```cpp
void loop()
{
    int buttonState = digitalRead(5);
    if(buttonState == LOW)
    {
        // do something
    }
    else
    {
        // do something else
    }
}
```
Analog Input

- Sample rate
- Bits per sample
• **Resolution**: the number of different voltage levels (i.e., states) used to represent an input signal

• Resolution values range from 256 states (8 bits) to 4,294,967,296 states (32 bits)

• The Arduino uses 1024 states (10 bits)

• Smallest measurable voltage change is 5V/1024 or 4.8 mV

• Maximum sample rate is 10,000 times a second
analogRead()

Arduino uses a 10-bit A/D Converter:
• this means that you get input values from 0 to 1023
  • 0 V → 0
  • 5 V → 1023

Ex:
  int sensorValue = analogRead(A0);
Serial Communication

Information passes between the computer and Arduino through the USB cable. Information is transmitted as zeros (‘0’) and ones (‘1’)… also known as **bits**!
Serial Communications

- “Serial” because data is broken down into bits, each sent one after the other down a single wire.

- The single ASCII character ‘B’ is sent as:

  \[
  \text{‘B’} = 0 1 0 0 0 0 0 1 0
  \]

  \[
  = \text{L H L L L L L H L}
  \]

  \[
  = \text{HIGH LOW}
  \]

- Toggle a pin to send data, just like blinking an LED

- You could implement sending serial data with `digitalWrite()` and `delay()`

- A single data wire needed to send data. One other to receive.
Serial Communication

- **Compiling** turns your program into binary data (ones and zeros)
- **Uploading** sends the bits through USB cable to the Arduino
- The two LEDs near the USB connector blink when data is transmitted
  - **RX** blinks when the Arduino is receiving data
  - **TX** blinks when the Arduino is transmitting data
Open the Serial Monitor and Upload the Program

/*
 * Hello World!
 * This is the Hello World! for Arduino.
 * It shows how to send data to the computer
 */

9600 baud

Hello world!
Some Commands

- Serial.begin()
  - e.g., Serial.begin(9600)
- Serial.print() or Serial.printLn()
  - e.g., Serial.print(value)
- Serial.read()
- Serial.available()
- Serial.write()
- Serial.parseInt()
Serial-to-USB chip---what does it do?

The LilyPad and Fio Arduino require an external USB to TTY connector, such as an FTDI “cable”. In the Arduino Leonardo a single microcontroller runs the Arduino programs and handles the USB connection.
Two different communication protocols

Serial (TTL):

If the Baud Rate = 9600 bps, then the Time/Bit = 1/9600 s
USB Protocol

Voltage signal in the differential pair

Differential decoding

NRZI decoding

Packet format

Start of packet / clock sync

Packet ID (LSB first, 1010 = NAK)

End of packet


more complicated
void loop ()
{
    Serial.print("Hands on ");
    Serial.print("Learning ");
    Serial.println("is Fun!!!");
}


void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.println("Hands on Learning is Fun!!!");
  //...
void loop() {
  int xVar = 10;
  Serial.print ( "Variable xVar is " ) ;
  Serial.println ( xVar ) ;
}
Serial Communication:
Serial Troubleshooting

```cpp
void loop ()
{
    Serial.print ("Digital pin 9: ");
    Serial.println (digitalRead(9));
}
```
Serial Monitor & analogRead()

```
void setup()
{
    Serial.begin(9600);
    pinMode(A0, INPUT);
}

void loop()
{
    sensorValue = analogRead(A0);
    Serial.println(sensorValue);
    delay(100); // waits by about 0.1 sec
}
```

- **Initialize the Serial Communication**
- 9600 baud data rate
- prints data to serial bus
Serial Monitor & analogRead()

 Opens up a Serial Terminal Window

```c
// analogRead() & Serial.print()
//
//

int sensorValue = 0;
int sensorPin = A0;

void setup()
{
    Serial.begin(9600);
    pinMode(A0, INPUT);
}

void loop()
{
    sensorValue = analogRead(A0);
    Serial.println(sensorValue);
    delay(100); // waits by about 0.1 sec
} 
```
Analog Sensors

2 Pin Analog Sensors = var. resistor

Take two sensors -- Use the Serial Monitor and find the range of input values you get for each sensor.

MaxAnalogRead = __________

MinAnalogRead = __________
Analog Sensors

Examples:

<table>
<thead>
<tr>
<th>Sensors</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mic</td>
<td>soundVolume</td>
</tr>
<tr>
<td>Photoresistor</td>
<td>lightLevel</td>
</tr>
<tr>
<td>Potentiometer</td>
<td>dialPosition</td>
</tr>
<tr>
<td>Temp Sensor</td>
<td>temperature</td>
</tr>
<tr>
<td>Flex Sensor</td>
<td>bend</td>
</tr>
<tr>
<td>Accelerometer</td>
<td>tilt/acceleration</td>
</tr>
</tbody>
</table>
Cell membrane

Receptor protein

Energy Stimulus

Sensory Receptor Cell

Signal transduction alters membrane permeability

Action Potential

Sensory Neuron

Vision
Taste
Smell
Hearing
Balance
Touch

Sensation and Reception
SENSE ORGANS

- Touch
- Pressure
- Temperature
- Skin

- Taste
- Tongue

- Ears
- Sound
- Balance

- Nose
- Smell
- Eyes
- Light
Telekinesis

Mind Reading

Levitation

Mind Control

Precognition

Remote Viewing

Mediumship

Energy Healing
Electret Microphone

- **Electret Membrane**: Maintains insulating gap
- **PICK-UP PLATE**: Forms capacitor with membrane
- **AMPLIFYING TRANSISTOR**: Silicon JFET N-channel type, with ground on source, pickup plate on gate, and signalout on drain
- **PLASTIC CASE**: Houses internal amplifier
- **PCB SUBSTRATE**: With holes for transistor leads
- **PCB TRACES**: Radiating "fingers" connect case to ground
- **CASE**: Crimped at back to retain contents
- **DUST COVER**: Cloth or paper
Ultrasonic Range Finder

- PING ultrasonic distance sensor provides precise distance measurements from about 2 cm (0.8 inches) to 3 meters (3.3 yards).

- It works by transmitting an ultrasonic burst and providing an output pulse that corresponds to the time required for the burst echo to return to the sensor.

- By measuring the echo pulse width the distance to target can easily be calculated.
Simple to Connect
- The PING sensor emits a short ultrasonic burst and then "listens" for the echo.
- Under control of a host microcontroller (trigger pulse), the sensor emits a short 40 kHz (ultrasonic) burst.
- This burst travels through the air at about 1130 feet per second, hits an object and then bounces back to the sensor.
- The PING sensor provides an output pulse to the host that will terminate when the echo is detected, hence the width of this pulse corresponds to the distance to the target.
Limited Detection Range