IME-100

ECE

Lab 4

Electrical and Computer Engineering Department

Kettering University

Getting Started

1. Laboratory Computers
   i. Log-in with User Name: Kettering Student (no password required)
   ii. IME-100 information (Lab presentation, files, etc.) in folder on desktop
   iii. Arduino programming software on desktop
   iv. At the end of lab, Logout of computer; arrange keyboard and mouse
2. Laboratory kit
   i. Arduino board, circuit prototyping breadboard, electronic components: LEDs, resistors, pushbutton switch, potentiometer, light sensor.
   ii. At the end of lab, turn instrument power off, dismantle circuit, unplug Arduino USB from PC, neatly arrange leads on bench

IME-100, ECE Lab4
Arduino Analog Input & Output

In this laboratory exercise, you will do the following:

- Explain the difference between digital and analog signals
- Use Arduino Analog Input functions to read analog signals
- Read analog input from potentiometer
- Read analog input from light sensor
- Explain how Arduino generates analog outputs
- Control external devices using analog output
- Write program to Arduino microcontroller
- Work on exercises using analog input and output

Analog Signal

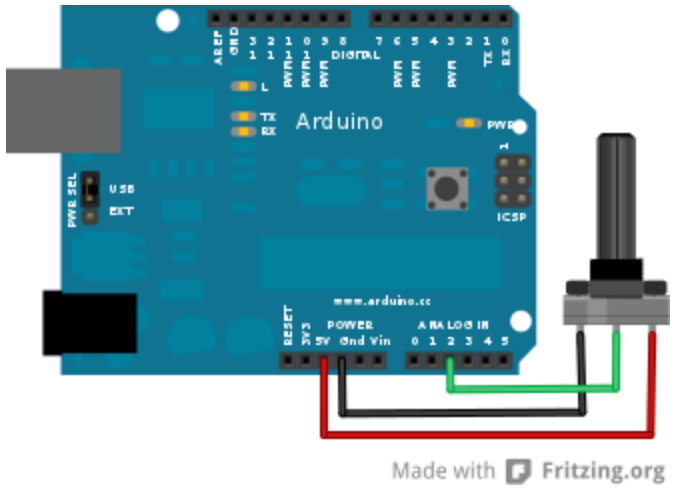
- ***Digital*** inputs such as those coming from switches (ON/OFF) could be handled by the Arduino without any conversion. Such digital signals are represented as HIGH and LOW, or 1 and 0.
  - Use ***digitalRead(pin)*** to access digital inputs to Arduino
- However, most of the things in nature are not in digital form. For example, if you want to build a weather station, your Arduino should be able to handle temperature, humidity, wind speed, rainfall, etc. The signals coming from these sensors are not restricted to HIGH and LOW (1 and 0, or ON/OFF), rather they can typically assume a wide range of possible values. This type of signals is called ***analog***.
- Arduino has a built in hardware called ***analog-to-digital convertor (ADC)*** that can be used for converting analog signals to digital format so that they can be processed by the Arduino.

Reading Analog Signal

- The ***analogRead*(analogPin)** function is used to convert an analog signal applied at one of the 6 analog pins of the Arduino Uno (A0, A1, … A5).
- You do not need to call ***pinMode()*** to set the pin as an input before calling ***analogRead()***
- The result of ***analogRead()*** is a 10-bit binary number in decimal range 0 to 1023.
- For example, if 2V analog signal is applied to pin A5, the ***analogRead(A5)*** function returns (2V/5V) *1023 = 409 (assuming a reference voltage 5V)

Potentiometer for Analog Signal

- A potentiometer is a device that provides a variable resistance, which can be converted to analog voltage for the Arduino to read.
- A potentiometer has three pins. Apply 5V and GND to the two end terminals, then connect the slider or blade terminal to the Arduino analog input pin to obtain a voltage that is proportional to the variable resistance.
- Potentiometer could be used for volume control, brightness control, and speed control knobs.
- As a simple example, the Arduino program in the next slide uses the potentiometer for controlling the blinking speed (or delay) of LED..



Displaying Data on PC Monitor

- Arduino provides Serial library to allow communication between the Arduino board and a computer or other devices.
- All Arduino boards have at least one serial port.
  - It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB.
  - Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.
- You can use the Arduino environment's built-in serial monitor to communicate with an Arduino board.
- Click the serial monitor button in the toolbar (top right) and select the same baud rate used in the call to begin().

Displaying Data on PC Monitor …

Example 1: Use of the serial monitor to display the data you receive from a potentiometer. Experiment and see how the data changes when you turn the potentiometer dial up and down.

```
/*
  AnalogReadSerial
 */

void setup() {
  // initialize serial communication
  //at 9600 bits per second:
  Serial.begin(9600);
}
```

```
void loop() {
  // read the input on analog pin 2:
  int sensorValue = analogRead(A2);

  // print out the value you read:
  Serial.println(sensorValue);

   // delay 1 sec between reads
  delay(1000);
}
```

Potentiometer for Analog Signal

Example 2: Use of the potentiometer to control the flashing of an LED. Assume the potentiometer is connected to the analog input pin A2. The LED on the Arduino board, connected to digital pin 13 could be used for the output.

```
// input pin for the potentiometer
int anPin = A2;
// output pin for the LED
int ledPin = 13;
// variable to store the analog voltage
int anInput = 0;

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
}
```

```
void loop() {
  // read the value from the sensor:
  anInput = analogRead(anPin);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // control delay with analog input value
  delay(anInput);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // control delay with analog input value
  delay(anInput);
}
```

Exercise 1: Use Potentiometer to Control timing of Traffic Light

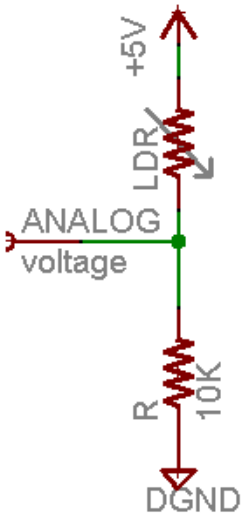
- For the traffic light circuit you built in a previous exercise, add a potentiometer to control the timing of the signals.
- You need to set the timing for the green and red signals to be proportional to the analog input value. For example, if the analog input is 100, set the duration of each of the red and green signals to 1 second. If the analog input is 500, set the red and green signal durations to 5 second, and so on. The yellow signal will always have a constant duration of 1 second.
- You will still need to provide the override feature as before using the pushbutton for control.
- Complete your Arduino program and demonstrate to the instructor your working traffic light circuit with the timing control using potentiometer, and with an override feature using the pushbutton switch.

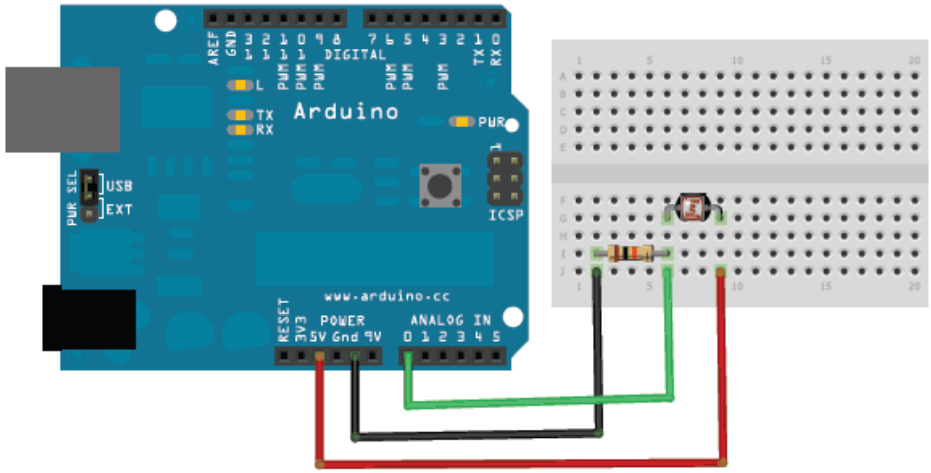
Light Sensor
Basic Operation

- We will use a photoresistor as a light sensor
- A photoresistor's ***resistance*** to electrical current changes with the intensity of light illumination. It is also called a light dependent resistance (LDR).
- A photoresistor is made of cadmium sulfide. It responds to visible light similarly to the human eye.
- Photoresistor resistance could range between about 200 KOhm in the dark and 500 ohm in a brightly lit room.
- Application: *light sensitive switches* – e.g. streetlights, vehicle headlights, garden lights, that automatically turn on at dusk

Light Sensor
Measuring Brightness

Photoresistor (LDR)





- Voltage reading is inversely proportional to light levels.
- Analog voltage reading ranges from 0 ~ 5V based on the LDR value
- The green wire is the point you read analog voltage from.
- Let us try to measure the voltage in various illuminations.
- In your experiment, you can use the light sensor provided to you and mounted in the front of your robot facing towards the floor.

Light Sensor
Display Sensor Reading on PC Monitor

Example 3: Use of the serial monitor to display the data you read from the light sensor. Experiment and see how the data changes when you change the light intensity by either covering the sensor with your hand or flashing a light (from your phone's flashlight) onto it.

```
#define lightPin  A0

void setup() {
 // initialize serial communication
 //at 9600 bits per second:
 Serial.begin(9600);
}
```

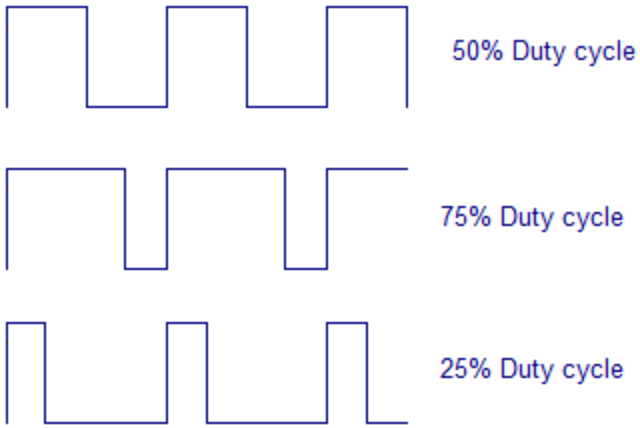
```
void loop() {
 // read the input on analog pin lightPin:
 int sensorValue = analogRead(lightPin);

 // print out the value you read:
 Serial.println(sensorValue);

  // delay 1 sec between reads
 delay(1000);
}
```

Exercise 2: Modify Traffic Light Operation at Night

- For the traffic light circuit you built in Exercise 1, add a light sensor that will be used for controlling the operating mode of the traffic light signals.
- First “calibrate” the light sensor by taking several sensor measurements under varying lighting intensities.
- Find a “threshold” value that splits the light intensity measurements into two, “dark” readings and “light” readings.
- Implement an Arduino program that controls the operation of the traffic signal as follows:
  - During the day time, the traffic signal operates in the “normal” operation mode; i.e. “green” (for 4 seconds), “yellow” (for 1 seconds), then “red” (for 4 seconds).
  - During the night time, the traffic signal only flashes the “red” light every one second; i.e. one second on and one second off.
- Complete your Arduino program and demonstrate to the instructor your working traffic light circuit with the required behavior.

Analog Output

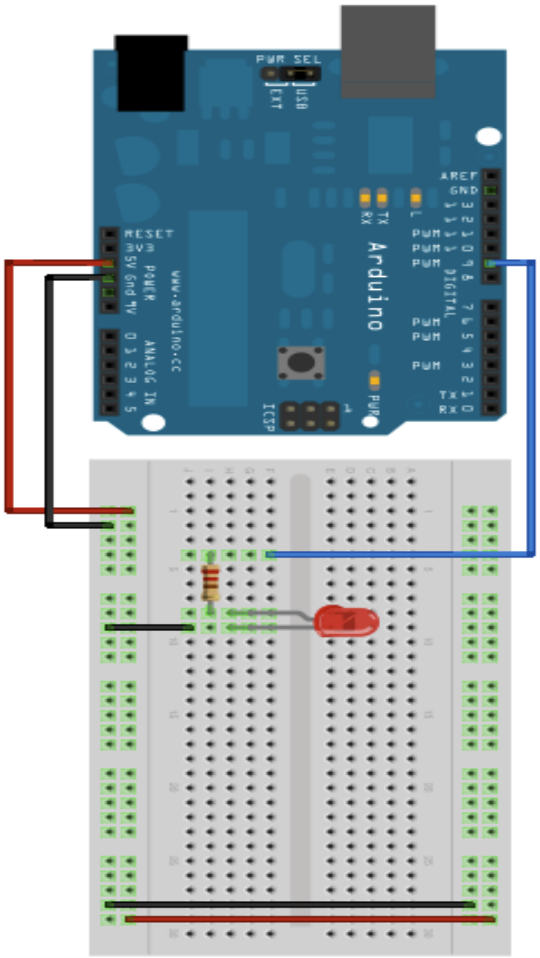
- The Arduino Uno does not actually support a 'real' ***analog output*** signal. But it can emulate analog output using ***pulse width modulation (PWM)***.
- ***PWM*** operates by turning a digital pin on and off very quickly in a periodic fashion. ***Duty cycle*** is the percentage of the time the signal stays HIGH relative to the period.



- PWM based analog outputs are available on pins 3, 5, 6, 9, 10, and 11 of the Arduino Uno. They are denoted by the ~ symbol as a prefix to the pin numbers on the board. Note that these are different from the analog input ports.
- ***analogRead(analogPin)*** function gets its input from the given analog pin.
- ***analogWrite(PWMpin,value)*** sends a PWM signal with a duty cycle in the range between 0 (always off) and 255 (always on), on the specified pin.

Analog Output

Example 3: Use the *analogWrite()* function to control LED brightness and demonstrate a fading effect




```
int led = 9;          // the pin that the LED is attached to
int brightness = 0;   // how bright the LED is
int fadeAmount = 5;   // how many points to fade the LED by

// the setup routine runs once when you press reset:
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);
 }
```

Exercise 3: Smart Street Light

- Use a bright “white” LED to emulate a street light.
- Make the street light ‘smart’ by using a light sensor as feedback.
- Control the ‘smart’ street light in such a way that the brightness of the street light is adjusted based on the measurements from the sensor readings.
  - Therefore, during the day the smart street light is completely off.
  - As the sun goes down, the smart street light gradually turns itself on at intensity levels that are appropriate to give sufficient lighting to the environment.
  - When it is completely dark, the ‘smart’ street light gives off its brightest light.
- Compare and contrast the ‘smart’ street light to a regular street light, based on energy efficiency and other factors.

Homework: Making Connections and For the Curious You …

Due: Beginning of 5th Week Lab

Making Connections

You are now able to design and implement different types of applications using a microcontroller, such as the Arduino and additional analog and digital devices such as sensors and switches. Research and identify an existing product or application that has an analog or digital sensor, a microcontroller and an output device that it controls. Explain each component of the product or application.

For the Curious You

Research and brainstorm/painstorm with your lab partners and present ***innovative ways of improving specific existing products/services by using a microcontroller, sensors and actuators***. For extra credit opportunity propose new a product/service with good analysis on its market potential.

Turn in a 2-3 pages report for your answers.

Homework: Online tutorial

From the online tutorial and additional resources come prepared to the next week class with background information on the following topics:

- Interfacing sensors
- Motor control
- Bluetooth communication

Next week:

- Use a motor control circuit to drive the robot
- Basic robot control
- Robotic sensors
- Use ultrasonic sensor to avoid crashes
- Light sensor and line following algorithm

Finishing Up

(and to get full-credit in the lab)

1. Clean-up at bench – Leave it better than you found it!
    i. Pick-up any spare parts, wire-trimmings, etc
    ii. Detangle and coil wire leads
    iii. Soldering stations and tools neatly arranged
    iv. Turn off instrument power, arrange neatly
    v. Logout of computer; arrange keyboard and mouse
    vi. Neatly arrange the chairs
2. Check-out with the instructor
    i. Leave the check-out sheet with your group names at your station

Lab 4 Check-Out Sheet

(to be left on the bench at the end of lab)

Group Members (please print name clearly):

_____

_____

_____

Instructor (check all that apply):

☐ **Exercise 1 demo**

☐ **Exercise 2 demo**

☐ **Exercise 3 demo**

☐ **Print-outs of program codes for all three exercises. Include group member names in comment at the top of your program file.**

☐ Computer Logout

☐ Bench clean-up

Wires, detangled and coiled,
Disposal of wire clipping, etc.
Arduino circuit dismantled
Arduino unplugged from PC USB
Instrument power off and arranged
Keyboard and Mouse arranged
Chairs arranged

Additional Comments: