6 Arduino

Objective

This lab introduces the Arduino microcontroller and the Arduino programming language. Arduino is a series of microcontrollers which can be used for to create custom scientific instruments and hobby projects, alike. Arduino is able to read inputs from various sensors and drive circuitry. In this lab, we will introduce the programming language through three examples: blinking an LED, fading and LED, and controlling a fading LED with a potentiometer.

Theory

Arduino was originally develped at Ivera Interaction Design Institute as a tool to bridge the gap between technology and art. The team designed Arduino to be cheaper than other microcontrollers and built an easy-to-use programming interface, which effectively lowered the barriers of entry for many different user groups.

The Arduino device is comprised of a microcontroller and various peripherals which feed the microcontroller inputs and outputs. Arduino can use both digital and analog inputs using the on-board analog-to-digital converter. Outputs may also be digital or "analog" using the pulse width modulation (PWM). PWM is a method of creating a rectangular wave with varying duty cycle and delivering an average power that can vary from 0% to 100% of the maximum power delivery of the device. Arduino also has a USB connector which is used for programming and powering the device.

There are many different types of Arduinos, each with their own features and utility. In this lab, we will focus on the Arduino Uno. The Arduino Uno is equiped with four onboard LEDs labeled: L, Rx, Tx, and ON. The Rx and Tx LEDs indicate when there is serial activity ocuring on the device. The ON LED indicates when the device is powered on. And the L LED is user-programmable and is tied directly to Pin 13 on the device. In other words, When Pin 13 is set to "1" or "High" the L LED will illuminate. The Arduino Uno is also equipped with a reset button which can be used to stop a program and restart it from the beginning.

Arduinos are programmed through the Arduino Integrated Development Environment (IDE) using the Arduino programming language. Programs are referred to as Sketches and the workspace within the IDE is called the Sketchbook. When you first open the IDE, the initial sketch will include two functions: setup() and loop(). The setup() function should be filled with commands that will only be run once during the start-up of the device. The setup() function is typically used for initializing the types of I/O the Arduino will be working with and initializing any peripheral devices that may communicate with the Arduino. The loop() function is used as the main run-time code. loop() will continuously run through the commands while the Arduino is powered on. The IDE also provides the user with a serial monitor which can receive serial messages from the Arduino. The serial monitor is useful when debugging a design and allows the user to print any message to the screen.

To get started with programming the Arduino Uno, we must select the correct board in the IDE. Start the Arduino IDE, then plug in the Arduino Uno. Select the "Tools" dropdown from the top navigation bar, then change the "Port" to the COM device the Arduino is connected through. Select "Tools" again and click "Get Board Info" to verify that it is communicating with the device. Click "Tools" one last time and ensure that the "Board" is set to "Arduino Uno"

Equipment

- Lab Computer
- Arduino Uno
- Breadboard
- LEDs
- Resistors

Procedure

6.1 Blink

The blink circuit is often treated as the "Hello World" for Arduino. We will program the Arduino to turn on and LED then turn it off again, and repeat. The LED used will be the onboard LED labeled "L", which is directly tied to Pin 13.

Note that "//" are used to define comments in the code.

Write the following sketch into your Arduino IDE:

```
//Define the led pin number using an integer variable
int led_pin = 13;

void setup() {
  // set the led_pin as an output
  pinMode(led_pin, OUTPUT);
}

void loop() {
  // Set the value of led_pin to high, turns on the LED
  digitalWrite(led_pin, HIGH);

  // delay for a moment
  delay(1000); // milliseconds

  // Set the value of led_pin to low, turns off the LED
  digitalWrite(led_pin, LOW);

  // delay for a moment
  delay(1000); // milliseconds
}
```

When done, save the sketch. Then click the Checkmark button to verify the sketch can compile. If there are no issues, use the Right-arrow button to write the sketch to the Aruduino Uno. You should now see the on-board LED blinking at regular intervals. Change the values of the delay to something different and observe the result.

6.2 Fading Blink

In this exercise, we will enhance the blink circuit to fade-in and fade-out. This time, we cannot use the on-board LED, as it is not connected to a PWM-enabled output pin. PWM pins are labeled with the ~ (tilde) character.

1. Obtain a breadboard, LED, and 220 Ω resistor
2. Connect the LED and resistor in series on the breadboard.
3. Use wire to connect Pin 3 on the Arduino to the Anode (positive lead) of the LED
4. Use wire to connect resistor to GND Pin on the Arduino

At this stage, you could change your Blink sketch to use Pin 3 instead of 13, and it will work with this external LED.

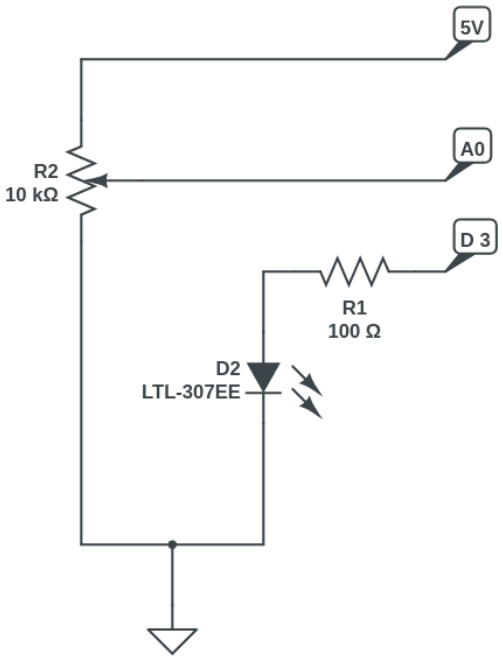
Next, let's write the sketch:

```
int led_pin = 3;
int brightness = 0;
int fade_amount = 5;

void setup() {
  // declare the led pin as output
  pinMode(led_pin, OUTPUT);

}

void loop() {
  // initilize the LED, analogWrite is used to control PWM
  // The second value can be any value from 0 to 255 to control the duty cycle
  // 0 = 0% duty cycle (off)
  // 255 = 100% duty cycle (on)
  analogWrite(led_pin, brightness);

  // change the brighness value using the fade_amount parameter
  brightness = brightness + fade_amount;

  // when brighness is full or off, reverse fade amount
  // If brightness is 0 OR brightness is 255, then negate the fade_amount.
  if (brightness == 0 || brightness == 255) {
    fade_amount = -fade_amount;
  }

  delay(50); //milliseconds

}
```

6.3 Controlled faded blink

In this section, we will use the a potentiometer to control the brightness level of the LED. A potentiometer is a resistor whose resistance can be adjusted by turning the dial on the potentiometer device. The change in resistance changes the voltage over the potentiometer according to voltage divider rule, which can be measured by the Arduino analog input.

Build the circuit below using a breadboard.



Write the sketch and upload to the Arduino.

```
int led_pin = 3;
int brightness = 0;
int fade_amount = 5;


void setup() {
  //declare the led_pin as OUTPUT
  pinMode(led_pin, OUTPUT);

  // Start the serial monitor at 9600 Baudrate
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
```

```
  // Read from the analog pin A0
  // Will read value from 0 to 1023
  int pot_sense = analogRead(A0);

  // print the potentiometer value to the serial monitor
  Serial.print("pot:");
  Serial.print(pot_sense);
  Serial.print(",");

  // Use the pot value to scale the brightness level between 0 and 255
  brightness = map(pot_sense, 0, 1023, 0, 255);

  // print the brightness value
  Serial.print("brightness:");
  Serial.println(brightness);

  // Write brightness value to led as PWM
  analogWrite(led_pin, pot_sense);

  // delay for 50 milliseconds
  delay(50);

}
```

With the sketch running and the Arduino connected to the PC, open the "Serial Monitor". Observe the comma-separated data being printed. Next, open the "Serial Plotter". Turn the dial on the potentiometer and observe the changes on plot. **Take a screenshot of the running Serial Plotter for inclusion in your lab report.**

Questions

1. Write a line of code that would declare pin 9 as an output which would be placed in the setup() function.

2. Explain the difference between the setup() function and the loop() function in an Arduino sketch.

3. The analogWrite() function uses pulse width modulation (PWM) to control the output power to the pin. What is the range of values that can be used, corresponding to 0% and 100% duty cycle?

4. In the third exercise, Section 6.3, we printed the data to the Serial monitor and used the Serial plotter to display it. Copy the code from the second exercise, Section 6.2, and augment it to print the brightness value to the Serial monitor so it could be used for plotting.