FLINT REGIONAL SCIENCE & ENGINEERING FAIR

Measurements, Sensors and Data Logging Course

Week 2

Office Hours

- Mondays @ 7:00 PM
  - Same Zoom link as our normal sessions

Lesson 2: Button

Use a pushbutton to change the state of the LED

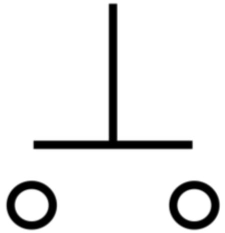
Digital Inputs!

Pushbutton Introduction

Lesson 2: Button

- What is a pushbutton?
  - A pushbutton is a momentary type of switch.
    - **Closed** = allows an electrical current to flow through it, completes the circuit
    - **Open** = prevents electrical current flow through it, circuit is not-complete
- Where are pushbuttons used?
  - Pushbuttons are used in many devices, from keyboards, cell phones, alarm clocks, industrial equipment, home appliances and much, much more.
  - Activity: find a device not listed above that uses a pushbutton.

Pushbutton Symbol



By Michel Bakni - Derived from files [1] and [2].(in English) (1993) 315-1975 - IEEE Standard American National Standard Canadian Standard Graphic Symbols for Electrical and Electronics Diagrams (Including Reference Designation Letters), IEEE, p. 59 DOI: 10.1109/IEEESTD.1993.93397. ISBN: 0738109479., CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=94264073

Example Pushbutton

Image from https://www.digikey.com/en/products/detail/e-switch/TL2230EEF140/4029358

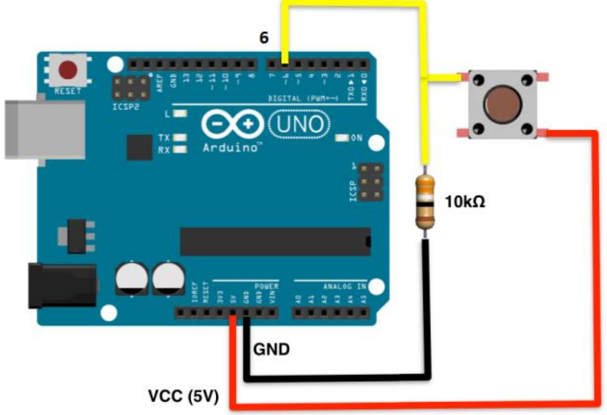
Pushbutton Introduction

Lesson 2: Button

- How do I use a pushbutton?
  - Follow the connection diagram to the right. Your Grove Beginner’s Kit has already done this for you.
  - We then read the state of the input using the digitalRead function.

- More Info:
  - https://www.allaboutcircuits.com/textbook/digital/chpt-4/switch-types/
  - https://en.wikipedia.org/wiki/Push-button

Example Pushbutton Connection



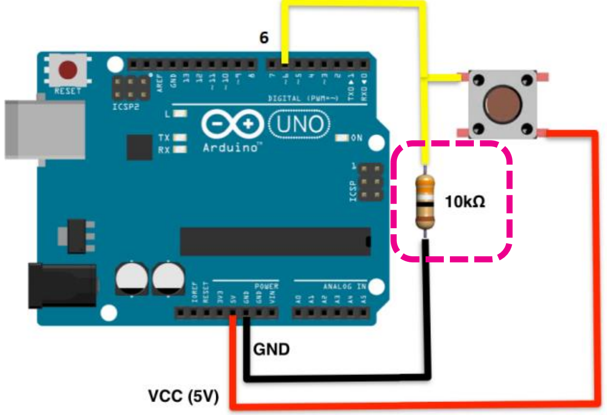
Modified from https://sites.google.com/site/ardunitydoc/getting-started/run-examples/push-button

Pull-Up / Pull-Down Resistors

Lesson 2: Button

- What happens if the circuit isn’t completed?
  - If the circuit is not completed (to GND or +5V), the input will “float”
  - To prevent the value from floating, we use a resistor to pull up or pull down the input.
    - 10K Ohm is a common pull up / down resistor value.
    - Pull Up = Resistor to Vcc (+5V)
    - Pull Down = Resistor to GND

Example Pushbutton Connection



Modified from https://sites.google.com/site/ardunitydoc/getting-started/run-examples/push-button

- More Info:
  - https://learn.sparkfun.com/tutorials/pull-up-resistors/all

Lesson 2 Hardware

Lesson 2: Button

- We can use our Arduino to receive a digital reading of the state of a pushbutton.
  - Receiving a **HIGH** signal means the button is **pressed**.
  - Receiving a **LOW** signal means the button is **released**.
- What hardware will we need for this Lesson?
  - Grove LED Module on pin D4
  - Grove Button Module on pin D6
  - Seeeduino Lotus (Arduino Uno compatible board)

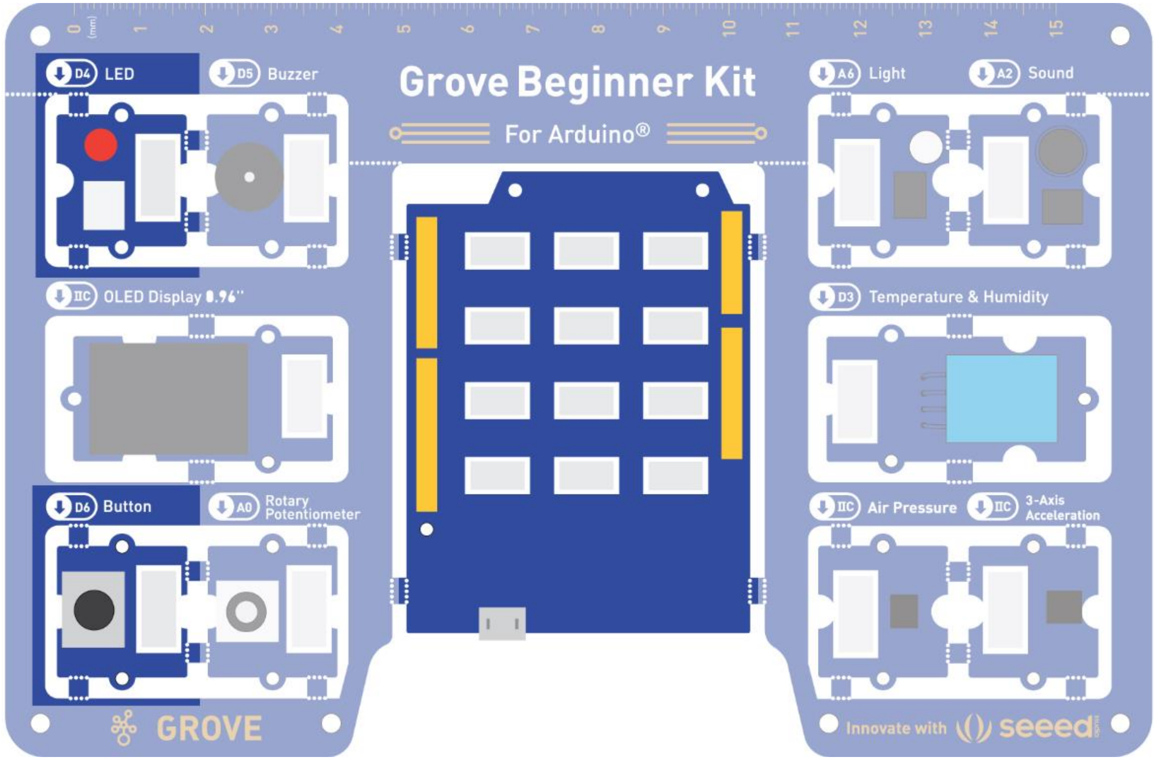


Image from https://files.seeedstudio.com/wiki/Grove-Beginner-Kit-For-Arduino/res/Grove-Beginner-Kit-For-ArduinoPDF.pdf

Open and Upload Sketch

Lesson 2: Button

1. Open Button Sketch
   a. **File → Sketchbook → FRSEF_Crash_Course → Week_ → W1L2_Button**
2. Verify the sketch by clicking the Verify Button.
   a. The sketch should compile with no errors.
3. Upload the sketch to your Arduino by clicking the Upload Button.
   a. The sketch should re-compile, and then upload to your Arduino.
4. Watch the LED as you press the button.

Code Analysis – pinMode Function

Lesson 2: Button

```
pinMode(buttonPin, INPUT);
```

- Configures the buttonPin as an input.

- The pinMode function configures the specified pin to behave either as an input or an output.
- Syntax:

```
    pinMode(pin, mode);
```

  - Pin: Arduino pin number to set the mode of
  - Mode: options are
    - `INPUT`, set pin as an input
    - `OUTPUT`, set pin as an output
    - `INPUT_PULLUP`, set pin as an input and enable a weak internal pullup resistor.
- More information:
  - https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode/

Code Analysis – Variable Assignment

Lesson 2: Button

```
buttonValue = digitalRead(buttonPin);
```

- Assigns the state of read buttonPin to buttonValue.

- The assignment operator (=) puts whatever is on the right side of the equal sign into the variable on the left side.
- Syntax:

```
    variable = value;
```

- Variable: stores the value of the statement on the right side of the equals sign.
- Value: statement, function or equation whose value is to be stored in variable.

- More information:
  - https://www.arduino.cc/en/Reference/VariableDeclaration

Code Analysis – if…else if…else Conditionals

Lesson 2: Button

- The **if** statement checks a condition and executes the proceeding statement(s) if the condition is TRUE.
- The **else** statement executes if the previous **if** conditional evaluated as FALSE. The **else** statement is optional.
- The **else if** statement combines the **else** statement with the **if** statement. The **else if** statement is optional.
- Syntax:

```
if(condition1)
{
   // do this
}
else if(condition2) // OPTIONAL
{
   // do that
}
else // OPTIONAL
{
   // do something else
}
```

  – **conditionX** must evaluate to TRUE (not 0) or FALSE (0)

- More information:
  – https://www.arduino.cc/reference/en/language/structure/control-structure/if/
  – https://www.arduino.cc/reference/en/language/structure/control-structure/else/

```
if(buttonValue == HIGH)
{
  digitalWrite(ledPin, HIGH);
}
else
{
  digitalWrite(ledPin, LOW);
}
```

Example of an if…else conditional

Button Activities

Lesson 2: Button

- Activity 1
  - Change the LED to turn OFF if the button is pressed and turn ON when the button is released.
- Activity 2 (Bonus / Homework)
  - Keep the same function as the original W1L2_Button.ino sketch, without using a conditional statement.
- Activity 3 (Bonus / Homework)
  - Toggle the LED every time the button is pressed.

Lesson 3: Potentiometer

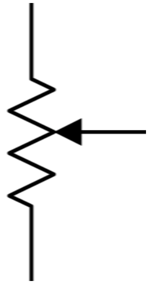
Use a potentiometer to change the brightness of the LED

Potentiometer Introduction

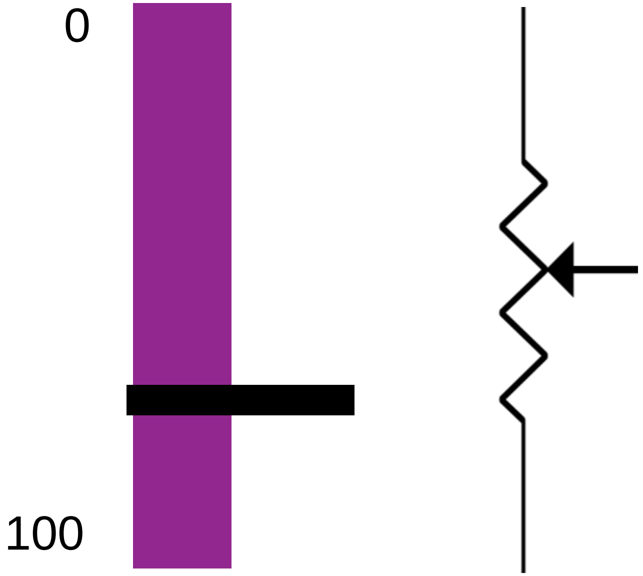
Lesson 3: Pot

- What is a potentiometer (pot)?
  - A pot is a type of variable resistor that has 3 terminals, two end terminals and a moveable wiper terminal.
  - Commonly used as position sensors.

Potentiometer Symbol



By Potentiometer with load.png: OmegatronDerivative work: DesbWit - This file was derived from: Potentiometer with load.png, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=20737313



Potentiometer Introduction

Lesson 3: Pot

- Where are pots used?
  - Pots are used in many devices, from volume knobs, industrial equipment, servos, home appliances, vehicles, and much, much more.
  - Activity: find a specific device that uses a pot.

Example Potentiometers

By Junkyardsparkle - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=39291275

Potentiometer Introduction

Lesson 3: Pot

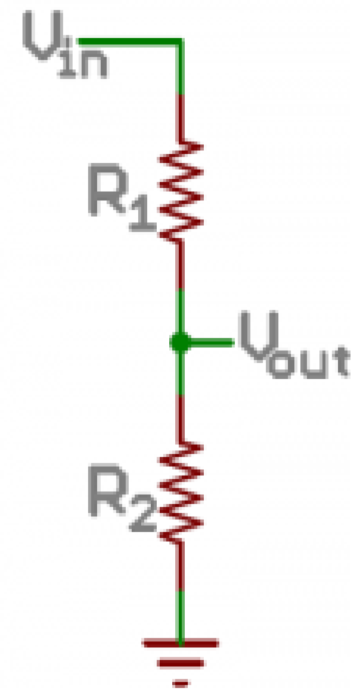
- What Is a Voltage Divider?
  - Simple circuit which turns a large voltage into a smaller one.
    - Vin = 5V
      - Ex. R1 = 50, R2 = 50, Vout = 2.5V

$$\text{Vout}= 5V * \frac{50}{50+50}$$

      - Ex. R1 = 20, R2 = 80, Vout = 4V

$$\text{Vout}= 5V * \frac{80}{20+80}$$

$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2}$$



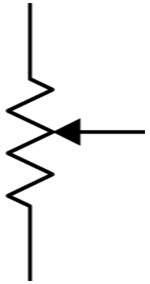
  - More Info:
    - https://learn.sparkfun.com/tutorials/voltage-dividers/all#:~:text=A%20voltage%20divider%20is%20a,most%20fundamental%20circuits%20in%20electronics

Potentiometer Introduction

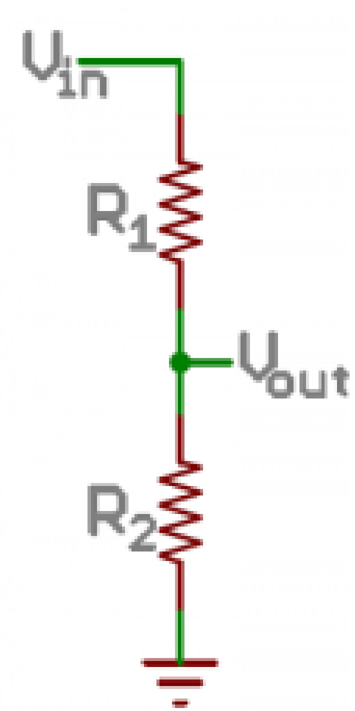
Lesson 3: Pot

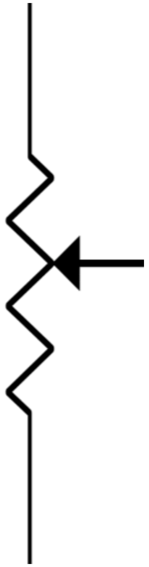
- Potentiometers are commonly used as adjustable voltage dividers.

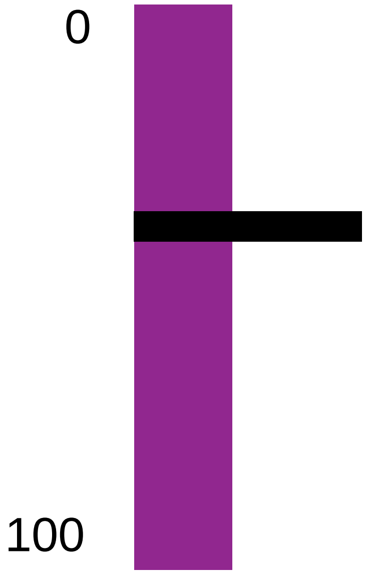
Potentiometer Symbol



By Potentiometer with load.png: OmegatronDerivative work: DesbWit - This file was derived from: Potentiometer with load.png, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=20737313







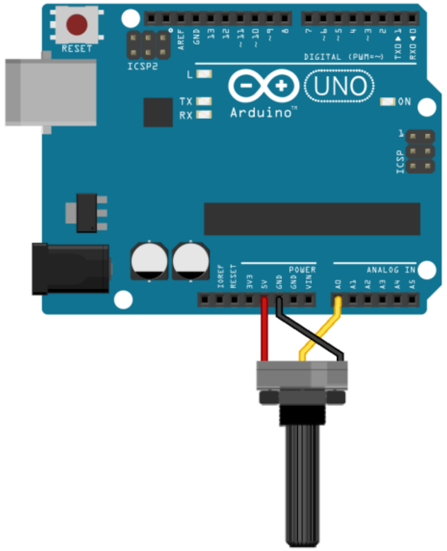
Potentiometer Introduction

Lesson 3: Pot

- How do I use a potentiometer?
  - Follow the connection diagram to the right. Your Grove Beginner's Kit has already done this for you.
  - We then read the state of the input using the analogRead function.

- More Info:
  - https://en.wikipedia.org/wiki/Potentiometer
  - https://www.allaboutcircuits.com/textbook/direct-current/chpt-6/voltage-divider-circuits/

Example Potentiometer Connection



Modified from https://www.arduino.cc/en/Tutorial/BuiltInExamples/AnalogInput

Pulse Width Modulation (PWM) Introduction

Lesson 3: Pot

- What is Pulse Width Modulation?
  - PWM is a type of digital signal that varies its value using the width of the pulse.
  - The value of the PWM signal is called the duty cycle (D). The duty cycle can be calculated as follows:
    - $\mathrm{D} = \frac{t_H}{t_H + t_L}$
    - Where t_H is the time the signal is high,
    - And t_L is the time the signal is low.

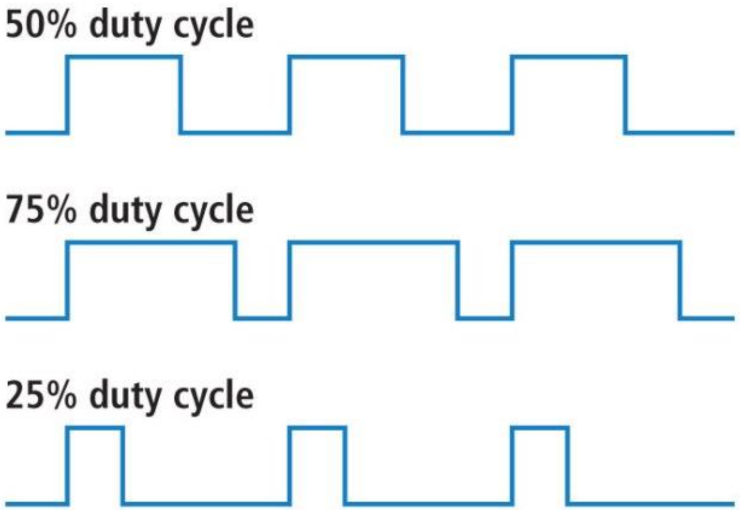
PWM Signal

By Eighthave - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=2821502

Pulse Width Modulation (PWM) Introduction

Lesson 3: Pot

- PWM Duty Cycle (D)
  - $\mathrm{D} = \frac{t_H}{t_H + t_L}$
    - Where t_H is the time the signal is high,
    - And t_L is the time the signal is low.
  - Calculate out of 100 ms (t_H + t_L = 100ms)
    - Duty Cycle = 50%
      - – t_H = 50
      - – $t_L = 50$
      - – $\mathrm{D} = \frac{50}{50+50}$
    - Duty Cycle = 75%
      - – t_H = ?
      - – t_L =?



Pulse Width Modulation (PWM) Introduction

Lesson 3: Pot

- Uses:
  - It is easy to convert a PWM signal back to an analog signal with a low pass filter.
  - We can use this to control the brightness of the LED.

- More Info:
  - https://en.wikipedia.org/wiki/Pulse-width_modulation
  - https://www.allaboutcircuits.com/textbook/semiconductors/chpt-11/pulse-width-modulation/

Combining Analog, PWM, and LEDs

Lesson 3: Pot

- We can use the MCU on our Arduino to read the value of the pot and output a PWM signal to the LED to control the brightness.
  - Outputting a higher value is a larger duty cycle which means a brighter LED.
- What hardware will we need for this Lesson?
  - Grove LED Module on pin D4
  - Grove Rotary Potentiometer Module on pin A0
  - Seeeduino Lotus (Arduino Uno compatible board)

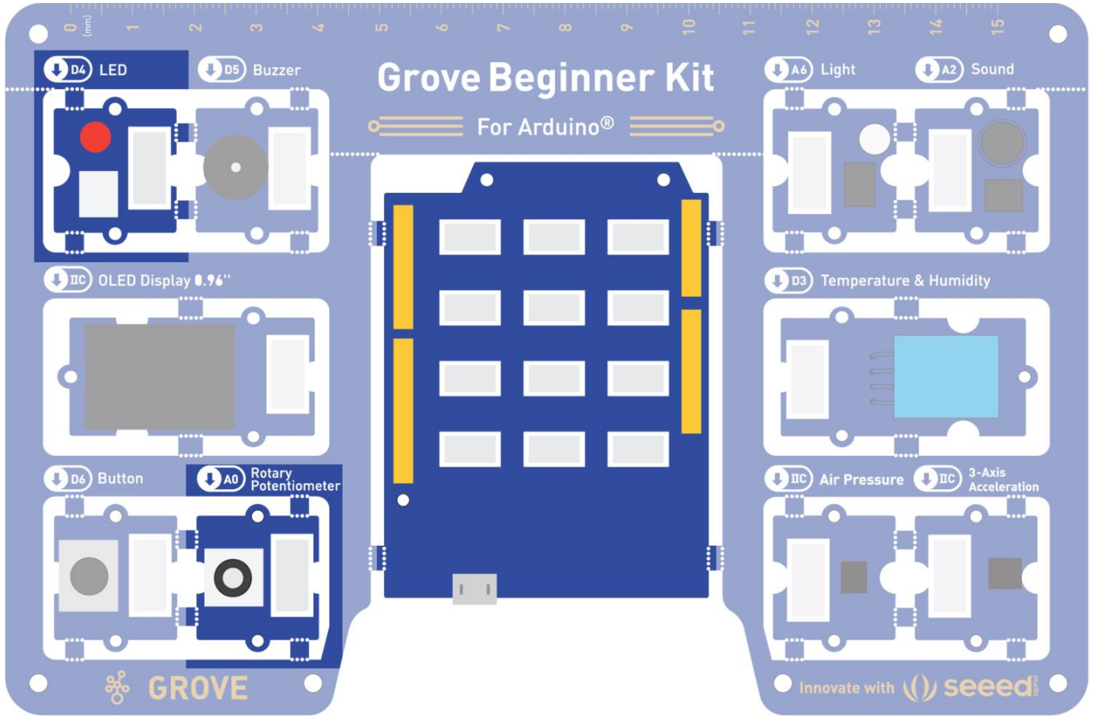


Image from https://files.seeedstudio.com/wiki/Grove-Beginner-Kit-For-Arduino/res/Grove-Beginner-Kit-For-ArduinoPDF.pdf

Open and Upload Sketch

Lesson 3: Pot

1. Open Pot Sketch
   a. **File → Sketchbook → FRSEF_Crash_Course → Week_ → W1L3_Pot**
2. Verify the sketch by clicking the Verify Button.
   a. The sketch should compile with no errors.
3. Upload the sketch to your Arduino by clicking the Upload Button.
   a. The sketch should re-compile, and then upload to your Arduino.
4. Watch the LED as you rotate the potentiometer.

Code Analysis – delayMicroseconds Function

Lesson 3: Pot

```
delayMicroseconds(potValue);
```

- Wait for the number of microseconds (µs) stored in potValue.

- This function is similar to the delay function from Lesson 1, except it pauses by microseconds instead if milliseconds.
- There are 1000µs in 1ms and 1,000,000µs in 1s.
- Syntax:

```
    delayMicroseconds(µs);
```

  - µs: number of microseconds (µs) to pause.
    - Data type is unsigned int with a range of 0 to 16,383µs (about 16 ms)
- More information:
  - https://www.arduino.cc/reference/en/language/functions/time/delaymicroseconds/

Pot Activities

Lesson 3: Pot

- Activity 1
  - Change the LED PWM to get brighter with a clockwise rotation of the potentiometer.
- Activity 2 (Bonus / Homework)
  - If the light sensor is on pin A6, modify the sketch to use the light sensor instead of the potentiometer.

Lesson 4: Light Sensor

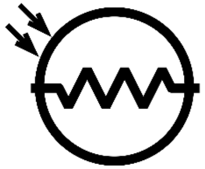
See the output of the light sensor in the Serial Monitor

Light Sensor Introduction

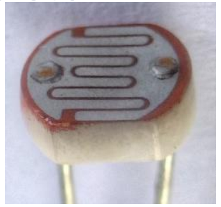
Lesson 4: Light Sensor

- What is a Light Sensor?
  - A light sensor is a type of device that changes a measurable electrical property based on the number (and type) of photons hitting it.
  - They come in many types but the main three for sensing applications are
    - **Photoresistors**: Resistance changes with light
    - **Photodiodes**: Photocurrent increases with light (this is also how a solar cell works)
    - **Phototransistors**: Amplified version of a photodiode.
- Where are light sensors used?
  - Occupancy sensors, daylight sensors, fiber optic communications, TVs (remote control receiver), cell phones, range finders, camera image sensors, etc.
  - Activity: Find a device not listed above that uses a light sensor.
- More information:
  - https://en.wikipedia.org/wiki/Photodetector
  - https://en.wikipedia.org/wiki/Photodiode
  - https://en.wikipedia.org/wiki/Photoresistor
  - https://www.seeedstudio.com/blog/2020/01/08/what-is-a-light-sensor-types-uses-arduino-guide/

Photoresistor

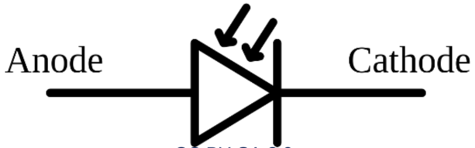


By User:FDominec et al. - File:Electrical_symbols_library.svg, CC0, https://commons.wikimedia.org/w/index.php?curid=49516462



By © Nevit Dilmen, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=30560805

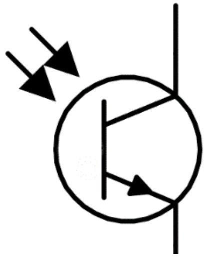
Photodiode



CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=755076

Copied from https://www.digikey.com/en/products/detail/w%C3%BCrth-elektronik/1540031EA4590/12366192

Phototransistor



By myself - WikiProject Wikipedia, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=32224508

Copied from https://www.digikey.com/en/products/detail/kingbright/WP7113P3C/7318904

Serial Introduction

Lesson 4: Light Sensor

- What is Serial Communication?
  - A digital signal where data is sent one bit at a time over a single channel.
  - Serial communications include RS232, RS485, UART, USART, USB, Ethernet, CAN, I^{2}C, SPI, SATA, etc.
  - Serial (without descriptors) typically refers to RS-232 and related communication signaling standards (UART or USART for a microcontroller).
- Where are serial communications used?
- More information:

Lesson 4 Hardware

Lesson 4: Light Sensor

- What hardware will we need for this Lesson?
  - Grove Light Sensor Module on pin A6
  - Seeeduino Lotus (Arduino Uno compatible board)
    - The Arduino has the serial port hardware built into the device

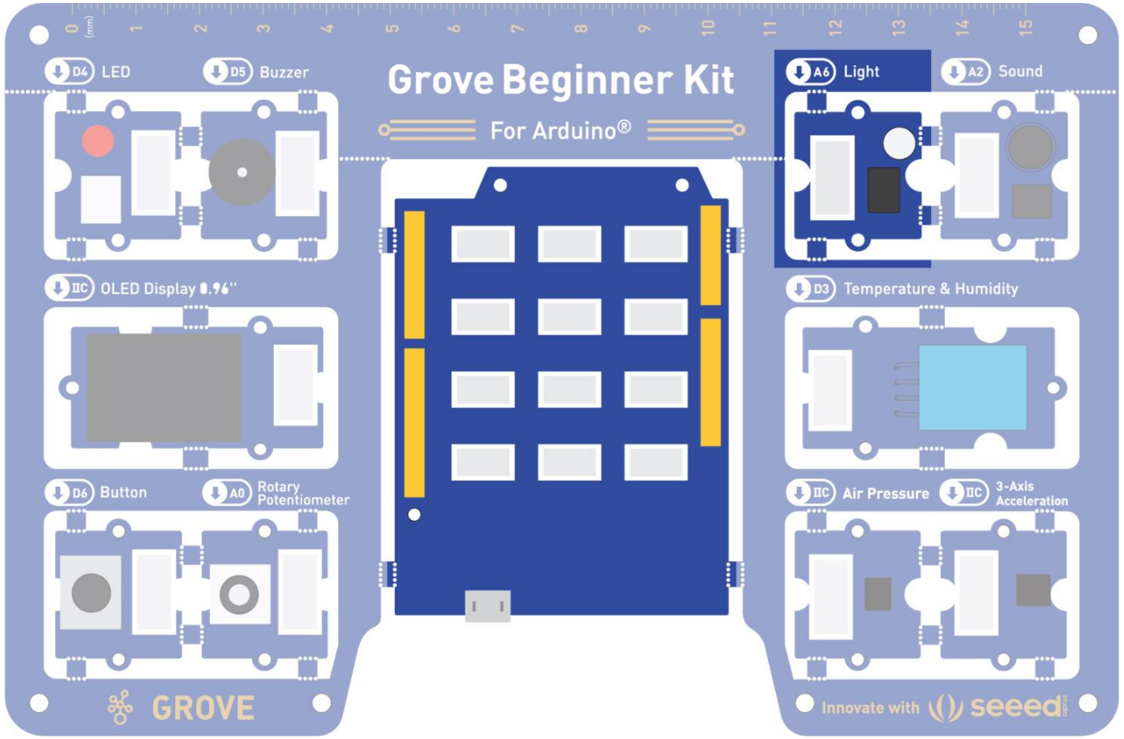
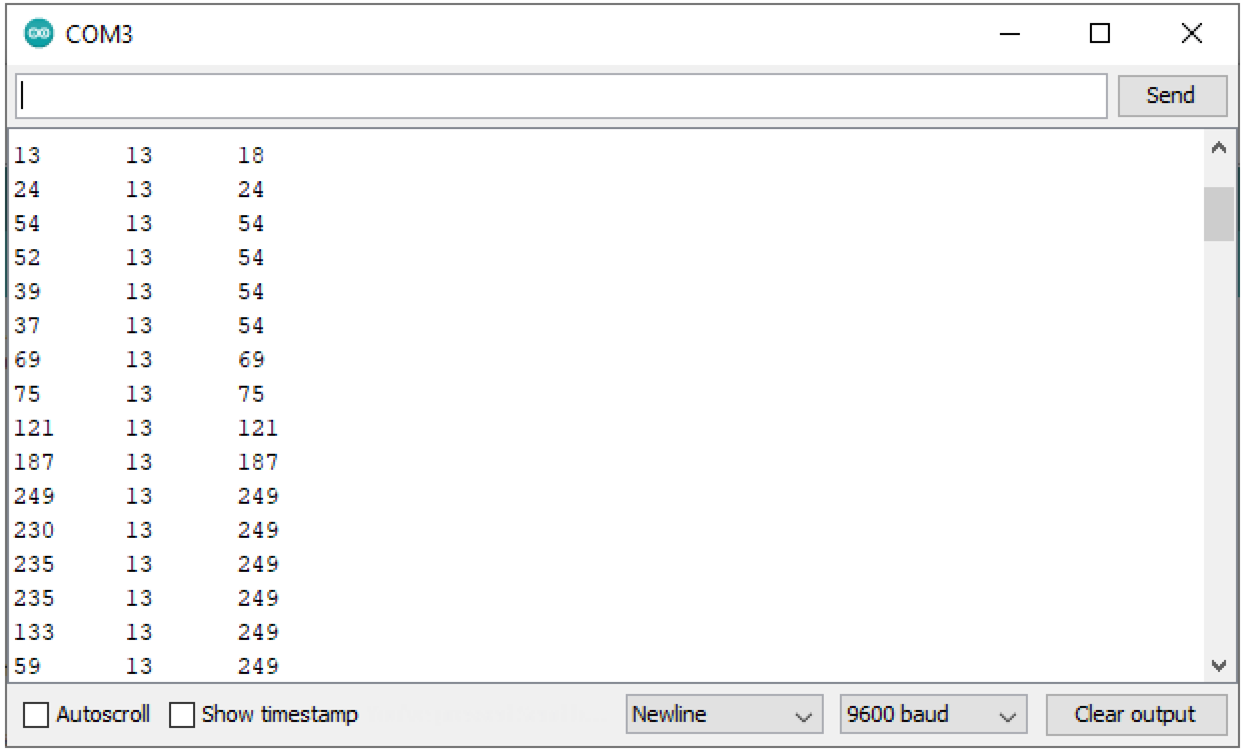


Image modified from https://files.seeedstudio.com/wiki/Grove-Beginner-Kit-For-Arduino/res/Grove-Beginner-Kit-For-ArduinoPDF.pdf

Open and Upload Sketch

Lesson 4: Light Sensor

1. Open Light_Serial Sketch
   a. **File → Sketchbook → FRSEF_Crash_Course → Week_2 → W2L4_Light_Serial.ino**
2. Verify the sketch by clicking the Verify Button.
   a. The sketch should compile with no errors.
3. Upload the sketch to your Arduino by clicking the Upload Button.
   a. The sketch should re-compile, and then upload to your Arduino.
4. Open the serial monitor.
   a. **Tools → Serial Monitor** (Ctrl+Shift+M)
5. Observe the output in the Serial Monitor



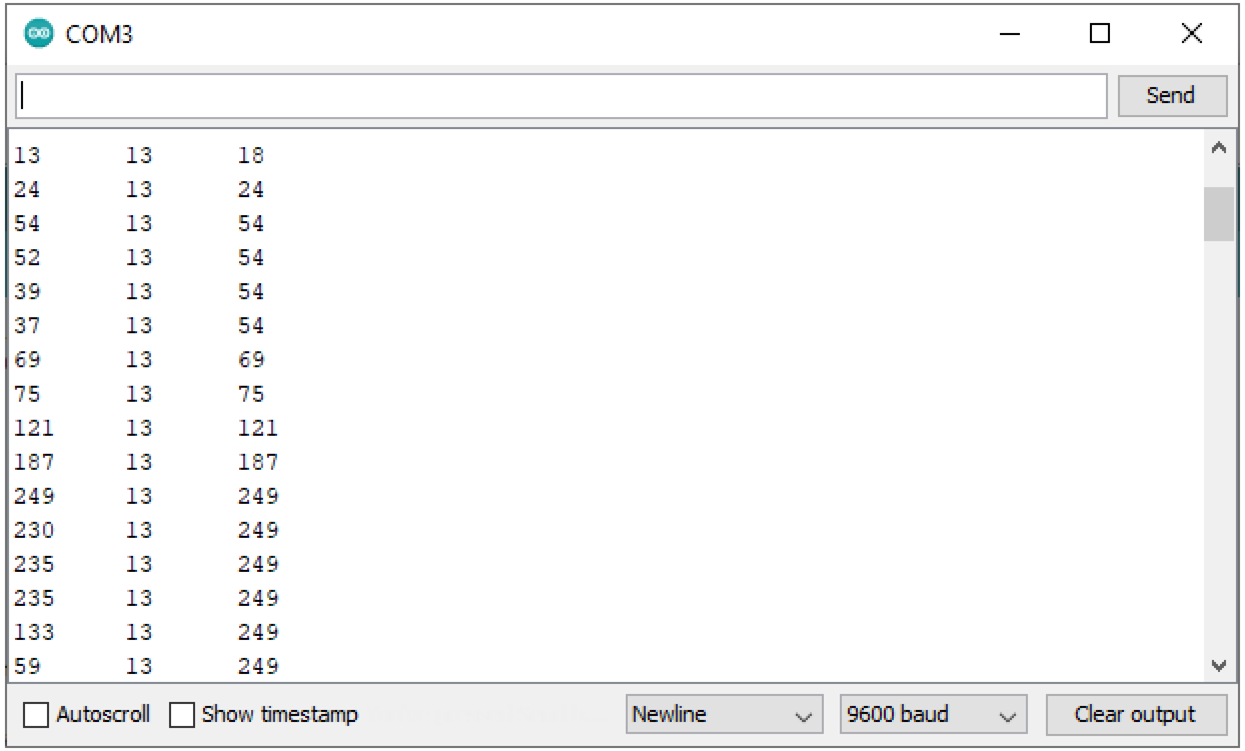
```
13      13      18
24      13      24
54      13      54
52      13      54
39      13      54
37      13      54
69      13      69
75      13      75
121     13      121
187     13      187
249     13      249
230     13      249
235     13      249
235     13      249
133     13      249
59      13      249
```

□ Autoscroll □ Show timestamp Newline 9600 baud

Serial Monitor

Lesson 4: Light Sensor

- What is the Serial Monitor?
  - The Serial Monitor is a feature of the Arduino IDE that gives you a serial terminal to see what is being sent to the COM port and allows you to send stuff out of the COM port.
  - We use this for receiving data from the Arduino.
  - We can also use this to help us debug our sketches.



```
13      13      18
24      13      24
54      13      54
52      13      54
39      13      54
37      13      54
69      13      69
75      13      75
121     13      121
187     13      187
249     13      249
230     13      249
235     13      249
235     13      249
133     13      249
59      13      249
```

□ Autoscroll □ Show timestamp

9600 baud

Code Analysis – `Serial` Functions

Lesson 4: Light Sensor

- `Serial.begin(9600);`
  - Start the Serial port at a 9600 baud
  - Put this function in the setup() function
  - Must call this function before using any other serial function
- `Serial.print(“string”);`
  - print a string or value to the serial port
- `Serial.println(“string”);`
  - same as print but add a new line character at the end of the string or value
- Special characters:
  - `‘\t’` is a Tab character
  - `‘\n’` is a New Line (some operating systems [⊞] use `“\r\n”`)
- More Information:
  - https://www.arduino.cc/reference/en/language/functions/communication/serial/
  - https://en.wikipedia.org/wiki/Control_character

Code Analysis – `min()` and `max()` Functions

Lesson 4: Light Sensor

`min(valueA, valueB);`

- Returns whichever value is lower

`max(valueA, valueB);`

- Returns whichever value is higher

- More information:
  - https://www.arduino.cc/reference/en/language/functions/math/max/
  - https://www.arduino.cc/reference/en/language/functions/math/min/

Light Sensor Activities

Lesson 4: Light Sensor

- Come up with some activity, or leave it for an overnight activity?

Review of Week 1 Activities

Blink Activity 1

Change the blink rate to 2Hz (2 blinks per second)

```
const int ledPin = 4; // Grove LED is on pin D4

void setup()
{
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT); // set the ledPin to be an output
}

void loop()
{
  // put your main code here, to run repeatedly:
  digitalWrite(ledPin, HIGH); // Turn the LED ON
  delay(250); // wait for 1/4 second
  digitalWrite(ledPin, LOW);  // Turn the LED OFF
  delay(250); // wait for 1/4 second
}
```

Blink Activity 2

Change the Pin to use the LED_BUILTIN keyword

```
const int ledPin = LED_BUILTIN; // Builtin LED

void setup()
{
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT); // set the ledPin to be an
output
}

void loop()
{
  // put your main code here, to run repeatedly:
  digitalWrite(ledPin, HIGH); // Turn the LED ON
  delay(1000); // wait for 1 second
  digitalWrite(ledPin, LOW);  // Turn the LED OFF
  delay(1000); // wait for 1 second
}
```

Blink Activity 3

Alternate LEDs that blink

```
const int ledPin = 4; // Grove LED is on pin D4
const int ledPin2 = LED_BUILTIN; // Builtin LED

void setup()
{
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT); // set the ledPin to be an output
  pinMode(ledPin2, OUTPUT); // set the ledPin2 to be an output
}

void loop()
{
  // put your main code here, to run repeatedly:
  digitalWrite(ledPin, HIGH); // Turn the Grove LED ON
  digitalWrite(ledPin2, LOW);  // Turn the Builtin LED OFF
  delay(1000); // wait for 1 second
  digitalWrite(ledPin, LOW);  // Turn the Grove LED OFF
  digitalWrite(ledPin2, HIGH); // Turn the Builtin LED ON
  delay(1000); // wait for 1 second
}
```

Blink Activity 4

Toggle the LED of your choice, without explicitly defining the state of the LED with HIGH or LOW

```
const int ledPin = 4; // Grove LED is on pin D4

void setup()
{
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT); // set the ledPin to be an output
}

void loop()
{
  // put your main code here, to run repeatedly:
  digitalWrite(ledPin, digitalRead(ledPin)); // Toggle the LED
  delay(1000); // wait for 1 second
}
```

Button Activity 1

Turn OFF LED when Button is pressed

```
const byte buttonPin = 6; // pushbutton is on D6
const byte ledPin = 4; // LED is on D4

byte buttonValue = 0; // global variable for storing the value of the button

void setup()
{
  pinMode(buttonPin, INPUT); // initialize the button as an input
  pinMode(ledPin, OUTPUT); // initialize the LED as an output
}

void loop()
{
  // read state of the button and store it in variable buttonValue
  buttonValue = digitalRead(buttonPin);

  if(buttonValue == LOW)  // Check if button is not pressed
  {
    digitalWrite(ledPin, HIGH); // Turn ON LED
  }
  else  // if it is not pressed
  {
    digitalWrite(ledPin, LOW);  // Turn OFF LED
  }
}
```

Button Activity 2

Turn ON LED when Button is pressed without using a conditional

```
const byte buttonPin = 6; // pushbutton is on D6
const byte ledPin = 4; // LED is on D4

byte buttonValue = 0; // global variable for storing the value of the button

void setup()
{
  pinMode(buttonPin, INPUT); // initialize the button as an input
  pinMode(ledPin, OUTPUT); // initialize the LED as an output
}

void loop()
{
  // read state of the button and output it to the LED
  digitalWrite(ledPin, digitalRead(buttonPin));
}
```

Button Activity 3

Toggle the LED when Button is pressed

```
const int buttonPin = 6; // pushbutton is on D6
const int ledPin = 4; // LED is on D4
const int debounce = 25; // ms for debounce delay

bool buttonState = LOW; // initialize button state

void setup()
{
  pinMode(buttonPin, INPUT); // initialize the button as an input
  pinMode(ledPin, OUTPUT); // initialize the led as an output
}

void loop()
{
  if((buttonState == LOW) && (digitalRead(buttonPin) == HIGH)); // detect button rising edge
  {
    digitalWrite(ledPin, not(digitalRead(ledPin))); // toggle ledPin
    delay(debounce); // allow for some button contact bouncing without triggering
    buttonState = HIGH; // Set buttonstate high to avoid constant triggering
  }
  if((buttonState == HIGH) && (digitalRead(buttonPin) == LOW)) // detect button falling edge
  {
    delay(debounce); // Allow for some contact bounce on release
    buttonState = LOW; // reset button press trigger
  }
}
```

Pot Activity 1

LED PWM gets brighter with clockwise rotation of pot

```
const byte ledPin = 4;  // LED is on pin D4
const byte potPin = A0; // Potentiometer is on pin A0
const int analogHigh = 1023;  // maximum analogRead value
unsigned int potValue = 0;

void setup()
{
  pinMode(ledPin, OUTPUT); // set the ledPin to be an output
}

void loop()
{
  potValue = analogRead(potPin);  //read the potentiometer
  // Create the PWM signal
  digitalWrite(ledPin, HIGH);  // Write the LED pin high
  delayMicroseconds((analogHigh - potValue) * 16);  // wait for the remainder of the period
  digitalWrite(ledPin, LOW); // Write the LED pin low
  delayMicroseconds(potValue * 16);  // delay by the pot value microseconds *16
}
```

Pot Activity 2

LED PWM uses the Light Sensor instead of the Pot

```
const byte ledPin = 4;   // LED is on pin D4
const byte lightPin = A6; // Light Sensor is on pin A6
const int analogHigh = 1023;   // maximum analogRead value
unsigned int lightValue = 0;

void setup()
{
  pinMode(ledPin, OUTPUT); // set the ledPin to be an output
}

void loop()
{
  lightValue = analogRead(lightPin);   //read the light sensor
  // Create the PWM signal
  digitalWrite(ledPin, HIGH);   // Write the LED pin high
  delayMicroseconds(lightValue * 16);   // delay by the lightValue microseconds *16
  digitalWrite(ledPin, LOW); // Write the LED pin low
  delayMicroseconds((analogHigh - lightValue) * 16);   // wait for the remainder of the period
}
```